MATRIXx™

Xmath™ Model Reduction Module

April 2004 Edition
Part Number 370755B-01

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 683 0100

Worldwide Offices

Australia 1800 300 800, Austria 43 0 662 45 79 90 0, Belgium 32 0 2 757 00 20, Brazil 55 11 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 450 510 3055,
Canada (Toronto) 905 785 0085, Canada (Vancouver) 514 685 7530, China 86 21 6555 7838,
Czech Republic 420 224 235 774, Denmark 45 45 76 26 00, Finland 385 0 9 725 725 11,
France 33 0 1 48 14 24 24, Germany 49 0 89 741 31 30, Greece 30 2 10 42 96 427, India 91 80 51190000,
Israel 972 0 3 6393737, Italy 39 02 413091, Japan 81 3 5472 2970, Korea 82 02 3451 3400,
Malaysia 603 9131 0918, Mexico 001 800 010 0793, Netherlands 31 0 348 433 466,
New Zealand 0800 553 322, Norway 47 0 66 90 76 60, Poland 48 22 3390150, Portugal 351 210 311 210,
Russia 7 095 783 68 51, Singapore 65 6226 5886, Slovenia 386 3 425 4200, South Africa 27 0 11 805 8197,
Spain 34 91 640 0085, Sweden 46 0 8 587 895 00, Switzerland 41 56 200 51 51, Taiwan 886 2 2528 7227,
Thailand 662 992 7519, United Kingdom 44 0 1635 523545

For further support information, refer to the *Technical Support and Professional Services* appendix. To comment on the documentation, send email to techpubs@ni.com.

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this document is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

MATRIXx™, National Instruments™, NI™, ni.com™, and Xmath™ are trademarks of National Instruments Corporation.

Product and company names mentioned herein are trademarks or trade names of their respective companies.

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Conventions

The following conventions are used in this manual:

[ ]	Square brackets enclose optional items—for example, [`response`]. Square brackets also cite bibliographic references.
»	The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box.
	This icon denotes a note, which alerts you to important information.
bold	Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names.
italic	Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.
`monospace`	Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.
`monospace bold`	Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.
`monospace italic`	Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Contents

Chapter 1
Introduction

Using This Manual 1-1
- Document Organization 1-1
- Bibliographic References 1-2
- Commonly Used Nomenclature 1-2
- Conventions 1-2
- Related Publications 1-3
- MATRIXx Help 1-3

Overview 1-3
- Functions 1-4
- Nomenclature 1-6

Commonly Used Concepts 1-7
- Controllability and Observability Grammians 1-7
- Hankel Singular Values 1-8
- Internally Balanced Realizations 1-10
- Singular Perturbation 1-11
- Spectral Factorization 1-13

Low Order Controller Design Through Order Reduction 1-15

Chapter 2
Additive Error Reduction

Introduction 2-1
Truncation of Balanced Realizations 2-2
Reduction Through Balanced Realization Truncation 2-4
Singular Perturbation of Balanced Realization 2-5
Hankel Norm Approximation 2-6
balmoore( ) 2-8
- Related Functions 2-11

truncate( ) 2-11
- Related Functions 2-11

redschur( ) 2-12
- Algorithm 2-12
- Related Functions 2-14

ophank( ) 2-14
- Restriction 2-14
- Algorithm 2-15
- Behaviors 2-15

Onepass Algorithm 2-18
Multipass Algorithm 2-20
Discrete-Time Systems 2-21
Impulse Response Error 2-22
Unstable System Approximation 2-23
Related Functions 2-23

Chapter 3
Multiplicative Error Reduction

Selecting Multiplicative Error Reduction 3-1
Multiplicative Robustness Result 3-2
bst( ) 3-3
Restrictions 3-4
Algorithm 3-4
Algorithm with the Keywords right and left 3-5
Securing Zero Error at DC 3-8
Hankel Singular Values of Phase Matrix of G_r 3-9
Further Error Bounds 3-9
Reduction of Minimum Phase, Unstable G 3-9
Imaginary Axis Zeros (Including Zeros at ∞) 3-10
Related Functions 3-14
mulhank( ) 3-14
Restrictions 3-14
Algorithm 3-14
right and left 3-15
Consequences of Step 5 and Justification of Step 6 3-18
Error Bounds 3-20
Imaginary Axis Zeros (Including Zeros at ∞) 3-21
Related Functions 3-24

Chapter 4
Frequency-Weighted Error Reduction

Introduction 4-1
Controller Reduction 4-2
Controller Robustness Result 4-2
Fractional Representations 4-5
wtbalance( ) 4-10
Algorithm 4-12
Related Functions 4-15
fracred( ) 4-15
Restrictions 4-15
Defining and Reducing a Controller 4-16

Algorithm 4-18
Additional Background 4-20
Related Functions 4-21

Chapter 5
Utilities

hankelsv( ) 5-1
Related Functions 5-2
stable( ) 5-2
Algorithm 5-2
Related Functions 5-4
compare( ) 5-4

Chapter 6
Tutorial

Plant and Full-Order Controller 6-1
Controller Reduction 6-5
ophank( ) 6-9
wtbalance 6-12
fracred 6-20

Appendix A
Bibliography

Appendix B
Technical Support and Professional Services

Index

1 Introduction

This chapter starts with an outline of the manual and some useful notes. It also provides an overview of the Model Reduction Module, describes the functions in this module, and introduces nomenclature and concepts used throughout this manual.

Using This Manual

This manual describes the Model Reduction Module (MRM), which provides a collection of tools for reducing the order of systems.

Readers who are not familiar with Parameter Dependent Matrices (PDMs) should consult the *Xmath User Guide* before using MRM functions and tools. Although several MRM functions accept both PDMs and matrices as input parameters, PDMs are preferable because they can include additional information that is useful for simulation, plotting, and signal labeling.

Document Organization

This manual includes the following chapters:

- Chapter 1, *Introduction*, starts with an outline of the manual and some useful notes. It also provides an overview of the Model Reduction Module, describes the functions in this module, and introduces nomenclature and concepts used throughout this manual.
- Chapter 2, *Additive Error Reduction*, describes additive error reduction including discussions of truncation of, reduction by, and perturbation of balanced realizations.
- Chapter 3, *Multiplicative Error Reduction*, describes multiplicative error reduction presenting considerations for using multiplicative rather than additive error reduction.
- Chapter 4, *Frequency-Weighted Error Reduction*, describes frequency-weighted error reduction problems, including controller reduction and fractional representations.

- Chapter 5, *Utilities*, describes three utility functions: `hankelsv( )`, `stable( )`, and `compare( )`.
- Chapter 6, *Tutorial*, illustrates a number of the MRM functions and their underlying ideas.

Bibliographic References

Throughout this document, bibliographic references are cited with bracketed entries. For example, a reference to [VODM1] corresponds to a paper published by Van Overschee and De Moor. For a table of bibliographic references, refer to Appendix A, *Bibliography*.

Commonly Used Nomenclature

This manual uses the following general nomenclature:

- Matrix variables are generally denoted with capital letters; vectors are represented in lowercase.
- $G(s)$ is used to denote a transfer function of a system where s is the Laplace variable. $G(q)$ is used when both continuous and discrete systems are allowed.
- $H(s)$ is used to denote the frequency response, over some range of frequencies of a system where s is the Laplace variable. $H(q)$ is used to indicate that the system can be continuous or discrete.
- A single apostrophe following a matrix variable, for example, x', denotes the transpose of that variable. An asterisk following a matrix variable, for example, A*, indicates the complex conjugate, or Hermitian, transpose of that variable.

Conventions

This publication makes use of the following types of conventions: font, format, symbol, mouse, and note. These conventions are detailed in Chapter 2, *MATRIXx Publications, Online Help, and Customer Support*, of the *MATRIXx Getting Started Guide*.

Related Publications

For a complete list of MATRIXx publications, refer to Chapter 2, *MATRIXx Publications, Online Help, and Customer Support*, of the *MATRIXx Getting Started Guide*. The following documents are particularly useful for topics covered in this manual:

- *MATRIXx Getting Started Guide*
- *Xmath User Guide*
- *Control Design Module*
- *Interactive Control Design Module*
- *Interactive System Identification Module, Part 1*
- *Interactive System Identification Module, Part 2*
- *Model Reduction Module*
- *Optimization Module*
- *Robust Control Module*
- *Xμ Module*

MATRIXx Help

Model Reduction Module function reference information is available in the *MATRIXx Help*. The *MATRIXx Help* includes all Model Reduction functions. Each topic explains a function's inputs, outputs, and keywords in detail. Refer to Chapter 2, *MATRIXx Publications, Online Help, and Customer Support*, of the *MATRIXx Getting Started Guide* for complete instructions on using the help feature.

Overview

The Xmath Model Reduction Module (MRM) provides a collection of tools for reducing the order of systems. Many of the functions are based on state-of-the-art algorithms in conjunction with researchers at the Australian National University, who were responsible for the original development of some of the algorithms. A common theme throughout the module is the use of Hankel singular values and balanced realizations, although considerations of numerical accuracy often dictates whether these tools are used implicitly rather than explicitly. The tools are particularly suitable when, as generally here, quality of approximation is measured by closeness of frequency domain behavior.

As shown in Figure 1-1, functions are provided to handle four broad tasks:

- Model reduction with additive errors
- Model reduction with multiplicative errors
- Model reduction with frequency weighting of an additive error, including controller reduction
- Utility functions

Functions

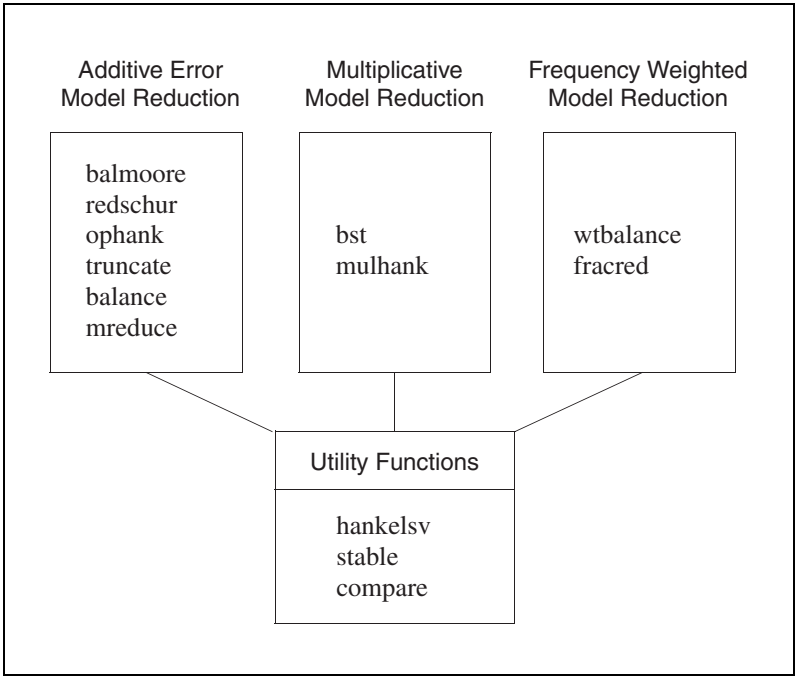

Figure 1-1. MRM Function

Certain restrictions regarding minimality and stability are required of the input data, and are summarized in Table 1-1.

Table 1-1. MRM Restrictions

`balance( )`	A stable, minimal system
`balmoore ( )`	A state-space system must be stable and minimal, having at least one input, output, and state
`bst( )`	A state-space system must be linear, continuous-time, and stable, with full rank along the $j\omega$-axis, including infinity
`compare( )`	Must be a state-space system
`fracred( )`	A state-space system must be linear and continuous
`hankelsv( )`	A system must be linear and stable
`mreduce( )`	A submatrix of a matrix must be nonsingular for continuous systems, and variant for discrete systems
`mulhank( )`	A state-space system must be linear, continuous-time, stable and square, with full rank along the $j\omega$-axis, including infinity
`ophank( )`	A state-space system must be linear, continuous-time and stable, but can be nonminimal
`redschur( )`	A state-space system must be stable and linear, but can be nonminimal
`stable ( )`	No restriction
`truncate( )`	Any full-order state-space system
`wtbalance( )`	A state-space system must be linear and continuous. Interconnection of controller and plant must be stable, and/or weight must be stable.

Documentation of the individual functions sometimes indicates how the restrictions can be circumvented. There are a number of model reduction methods not covered here. These include:

- Padé Approximation
- Methods based on interpolating, or matching at discrete frequencies

- L_2 approximation, in which the L_2 norm of impulse response error (or, by Parseval's theorem, the L_2 norm of the transfer-function error along the imaginary axis) serves as the error measure
- Markov parameter or impulse response matching, moment matching, covariance matching, and combinations of these, for example, q-COVER approximation
- Controller reduction using canonical interactions, balanced Riccati equations, and certain balanced controller reduction algorithms

Nomenclature

This manual uses standard nomenclature. The user should be familiar with the following:

- *sup* denotes supremum, the least upper bound.
- The acute accent (´) denotes matrix transposition.
- A superscripted asterisk (*) denotes matrix transposition and complex conjugation.
- $\lambda_{max}(A)$ for a square matrix A denotes the maximum eigenvalue, presuming there are no complex eigenvalues.
- $\mathrm{Re}\lambda_i(A)$ and $|\lambda_i(A)|$ for a square matrix A denote an arbitrary real part and an arbitrary magnitude of an eigenvalue of A.
- $\|X(j\omega)\|_\infty$ for a transfer function $X(\cdot)$ denotes:

$$\sup_{-\infty < \omega < \infty} [\lambda_{max}[X^*(j\omega)X(j\omega)]]^{1/2}$$

- An all-pass transfer-function $W(s)$ is one where $|X(j\omega)| = 1$ for all ω; to each pole, there corresponds a zero which is the reflection through the $j\omega$-axis of the pole, and there are no $j\omega$-axis poles.
- An all-pass transfer-function matrix $W(s)$ is a square matrix where

$$W'(-j\omega)W(j\omega) = I$$

  $P > 0$ and $P \geq 0$ for a symmetric or hermitian matrix denote positive and nonnegative definiteness.
- $P_1 > P_2$ and $P_1 \geq P_2$ for symmetric or hermitian P_1 and P_2 denote $P_1 - P_2$ is positive definite and nonnegative definite.
- A superscripted number sign (#) for a square matrix A denotes the Moore-Penrose pseudo-inverse of A.

- An inequality or bound is tight if it can be met in practice, for example

$$1 + \log x - x \leq 0$$

is tight because the inequality becomes an equality for $x = 1$. Again, if $F(j\omega)$ denotes the Fourier transform of some $f(t) \in L_2$, the Heisenberg inequality states,

$$\frac{\int |f(t)^2 dt|}{\left[\int t^2 |f(t)|^2 dt\right]^{1/2} \left[\int \omega^2 |F(j\omega)|^2 d\omega\right]^{1/2}} \leq 4\pi$$

and the bound is tight since it is attained for $f(t) = \exp + (-kt^2)$.

Commonly Used Concepts

This section outlines some frequently used standard concepts.

Controllability and Observability Grammians

Suppose that $G(s) = D + C(sI-A)^{-1}B$ is a transfer-function matrix with $Re\lambda_i(A)<0$. Then there exist symmetric matrices P, Q defined by:

$$PA' + AP = -BB'$$
$$QA + A'Q = -C'C$$

These are termed the controllability and observability grammians of the realization defined by $\{A,B,C,D\}$. (Sometimes in the code, *WC* is used for P and *WO* for Q.) They have a number of properties:

- $P \geq 0$, with $P > 0$ if and only if $[A,B]$ is controllable, $Q \geq 0$ with $Q > 0$ if and only if $[A,C]$ is observable.
- $P = \int_0^\infty e^{At} BB' e^{A't} dt$ and $Q = \int_0^\infty e^{A't} C'C e^{At} dt$
- With *vec P* denoting the column vector formed by stacking column 1 of P on column 2 on column 3, and so on, and $\otimes$ denoting Kronecker product

$$[I \otimes A + A \otimes I]\text{vec}P = -\text{vec}(BB')$$

- The controllability grammian can be thought of as measuring the difficulty of controlling a system. More specifically, if the system is in the zero state initially, the minimum energy (as measured by the L_2 norm of u) required to bring it to the state x_0 is $x_0 P^{-1} x_0$; so small eigenvalues of P correspond to systems that are difficult to control, while zero eigenvalues correspond to uncontrollable systems.

- The controllability grammian is also $E[x(t)x'(t)]$ when the system $\dot{x} = Ax + Bw$ has been excited from time $-\infty$ by zero mean white noise with $E[w(t)w'(s)] = I\delta(t-s)$.
- The observability grammian can be thought of as measuring the information contained in the output concerning an initial state. If $\dot{x} = Ax, y = Cx$ with $x(0) = x_0$ then:

$$\int_0^\infty y'(t)y(t)dt = x'_0 Q x_0$$

  Systems that are easy to observe correspond to Q with large eigenvalues, and thus large output energy (when unforced).
- `lyapunov(A,B*B')` produces P and `lyapunov(A',C'*C)` produces Q.

  For a discrete-time $G(z) = D + C(zI\text{-}A)^{-1}B$ with $|\lambda_i(A)|<1$, P and Q are:

$$P - APA' = BB'$$
$$Q - A'QA = C'C$$

  The first dot point above remains valid. Also,
- $P = \sum_{k=0}^{\infty} A^k BB' A'^k$ and $Q = \sum_{k=0}^{\infty} A^k C'CA'^k$

  with the sums being finite in case A is nilpotent (which is the case if the transfer-function matrix has a finite impulse response).
- $[I - A \otimes A]$ vec P = vec (BB')

  `lyapunov( )` can be used to evaluate P and Q.

Hankel Singular Values

If P, Q are the controllability and observability grammians of a transfer-function matrix (in continuous or discrete time), the *Hankel Singular Values* are the quantities $\lambda_i^{1/2}(PQ)$. Notice the following:

- All eigenvalues of PQ are nonnegative, and so are the Hankel singular values.
- The Hankel singular values are independent of the realization used to calculate them: when A,B,C,D are replaced by TAT^{-1}, TB, CT^{-1} and D, then P and Q are replaced by TPT' and $(T^{-1})'QT^{-1}$; then PQ is replaced by $TPQT^{-1}$ and the eigenvalues are unaltered.
- The number of nonzero Hankel singular values is the order or McMillan degree of the transfer-function matrix, or the state dimension in a minimal realization.

- Suppose the transfer-function matrix corresponds to a discrete-time system, with state variable dimension n. Then the infinite Hankel matrix,

$$H = \begin{bmatrix} CB & CAB & CA^2B \\ CAB & CA^2B & \\ CA^2B & & \end{bmatrix}$$

  has for its singular values the n nonzero Hankel singular values, together with an infinite number of zero singular values.

The Hankel singular values of a (stable) all pass system (or all pass matrix) are all 1.

Slightly different procedures are used for calculating the Hankel singular values (and so-called weighted Hankel singular values) in the various functions. These procedures are summarized in Table 1-2.

Table 1-2. Calculating Hankel Singular Values

`(balance( ))`	For a discussion of the balancing algorithm, refer to the *Internally Balanced Realizations* section; the Hankel singular values are given by $diag(R^{1/2}) = HSV$
`balmoore( )`	For a discussion of the balancing algorithm, refer to the *Internally Balanced Realizations* section; the matrix S_H yields the Hankel singular values through `diag(SH)`
`hankelsv( )`	`real(sqrt(eig(p*q)))`
`ophank( )`	Calls `hankelsv( )`
`redschur( )`	Computes a Schur decomposition of $P*Q$ and then takes the square roots of the diagonal entries

`bst( )`
`mulhank( )`
`wtbalance( )`
`fracred( )`
	Same as `redschur( )` except either P or Q can be a weighted grammian

Internally Balanced Realizations

Suppose that a realization of a transfer-function matrix has the controllability and observability grammian property that $P = Q = \Sigma$ for some diagonal Σ. Then the realization is termed internally balanced. Notice that the diagonal entries σ_i of Σ are square roots of the eigenvalues of PQ, that is, they are the Hankel singular values. Often the entries of Σ are assumed ordered with $\sigma_i \geq \sigma_{i+1}$.

As noted in the discussion of grammians, systems with small (eigenvalues of) P are hard to control and those with small (eigenvalues of) Q are hard to observe. Now a state transformation $T = \alpha I$ will cause $P = Q$ to be replaced by $\alpha^2 P$, $\alpha^{-2} Q$, implying that ease of control can be obtained at the expense of difficulty of observation, and conversely. Balanced realizations are those when ease of control has been balanced against ease of observation.

Given an arbitrary realization, there are a number of ways of finding a state-variable coordinate transformation bringing it to balanced form.

A good survey of the available algorithms for balancing is in [LHPW87]. One of these is implemented in the Xmath function `balance( )`.

The one implemented in `balmoore( )` as part of this module is more sophisticated, but more time consuming. It proceeds as follows:

1. Singular value decompositions of P and Q are defined. Because P and Q are symmetric, this is equivalent to diagonalizing P and Q by orthogonal transformations.

$$P = U_c S_c U'_c$$
$$Q = U_o S_o U'_o$$

2. The matrix,

$$H = S_0^{1/2} U_H S_H V_H^{1/2}$$

   is constructed, and from it, a singular value decomposition is obtained:

$$H = U_H S_H V_H^{'}$$

3. The balancing transformation is given by:

$$T = U_0 S_0^{-1/2} U_H S_H^{1/2}$$

   The balanced realization is $T^{-1}AT$, $T^{-1}B$, CT.

This is almost the algorithm set out in Section II of [LHPW87]. The one difference (and it is minor) is that in [LHPW87], lower triangular Cholesky factors of P and Q are used, in place of $U_c S_c^{1/2}$ and $U_O S_O^{1/2}$ in forming H in step 2. The grammians themselves need never be formed, as these Cholesky factors can be computed directly from A, B, and C in both continuous and discrete time; this, however, is not done in `balmoore`.

The algorithm has the property that:

$$T'QT' = T^{-1}P(T^{-1})' = S_H$$

Thus the diagonal entries of S_H are the Hankel singular values.

The algorithm implemented in `balance( )` is older, refer to [Lau80]. A lower triangular Cholesky factor L_c of P is found, so that $L_c L_c' = P$. Then the symmetric matrix $L_c'QL_c$ is diagonalized (through a singular value decomposition), thus $L'_c\mathrm{QL}_c = VRU'$, with actually $V = U$. Finally, the coordinate basis transformation is given by $T = L_c VR^{-1/4}$, resulting in $T'QT = T^{-1}P(T^{-1})' = R^{1/2}$.

Singular Perturbation

A common procedure for approximating systems is the technique of Singular Perturbation. The underlying heuristic reasoning is as follows. Suppose there is a system with the property that the unforced responses of the system contain some modes which decay to zero extremely fast. Then an approximation to the system behavior may be attained by setting state variable derivatives associated with these modes to zero, even in the forced case. The phrase “associated with the modes” is loose: exactly what occurs is shown below. The phrase “even in the forced case” captures a logical flaw in the argument: smallness in the unforced case associated with initial conditions is not identical with smallness in the forced case.

Suppose the system is defined by:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}B_1\\ B_2\end{bmatrix}u$$

$$y = \begin{bmatrix}C_1 & C_2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + Du \qquad (1\text{-}1)$$

and also:

$$Re\lambda_i(A_{22})<0 \quad \text{and} \quad Re\lambda_i(A_{11} - A_{12}A_{22}^{-1}A_{21}) < 0.$$

Usually, we expect that,

$$Re\lambda_i(A_{22}) \ll Re\lambda_i(A_{11} - A_{12}A_{22}^{-1}A_{21})$$

in the sense that the intuitive argument hinges on this, but it is not necessary.

Then a singular perturbation is obtained by replacing $\dot{x}_2$ by zero; this means that:

$$A_{21}x_1 + A_{22}x_2 + B_2u = 0 \qquad \text{or} \qquad x_2 = -A_{22}^{-1}A_{21}x_1 - A_{22}^{-1}B_2u$$

Accordingly,

$$\begin{aligned} \dot{x}_1 &= (A_{11} = A_{12}A_{22}^{-1}A_{21})x_1 + (B_1 - A_{12}A_{22}^{-1}B_2)u \\ y &= (C_1 - C_2A_{22}^{-1}A_{21})x_1 + (D - C_2A_{22}^{-1}B_2)u \end{aligned} \tag{1-2}$$

Equation 1-2 may be an approximation for Equation 1-1. This means that:

- The transfer-function matrices may be similar.
- If Equation 1-2 is excited by some $u(\cdot)$, with initial condition $x_1(t_o)$, and if Equation 1-1 is excited by the same $u(\cdot)$ with initial condition given by,
- $x_1(t_o)$ and $x_2(t_o) = -A^{-1}{}_{22}A_{21}x_1(t_o) - A_{22}{}^{-1}B_{2u}(t_o)$,

  then $x_1(\cdot)$ and $y(\cdot)$ computed from Equation 1-1 and from Equation 1-2 should be similar.
- If Equation 1-1 and Equation 1-2 are excited with the same $u(\cdot)$, have the same $x_1(t_o)$ and Equation 1-1 has arbitrary x_2, then $x_1(\cdot)$ and $y(\cdot)$ computed from Equation 1-1 and Equation 1-2 should be similar after a possible initial transient.

As far as the transfer-function matrices are concerned, it can be verified that they are actually equal at DC.

Similar considerations govern the discrete-time problem, where,

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + Du(k)$$

can be approximated by:

$$\begin{aligned} x_1(k+1) &= [A_{11} + A_{12}(I - A_{22})^{-1} A_{21}] x_1(k) + \\ &\quad [B_1 + A_{12}(I - A_{22})^{-1} B_2] u(k) \\ y_k &= [C_1 + C_2(I - A_{22})^{-1} A_{21}] x_1(k) + \\ &\quad [D + C_2(I - A_{22})^{-1} B_2] u(k) \end{aligned}$$

`mreduce( )` can carry out singular perturbation. For further discussion, refer to Chapter 2, *Additive Error Reduction*. If Equation 1-1 is balanced, singular perturbation is provably attractive.

Spectral Factorization

Let $W(s)$ be a stable transfer-function matrix, and suppose a system S with transfer-function matrix $W(s)$ is excited by zero mean unit intensity white noise. Then the output of S is a stationary process with a spectrum $\Phi(s)$ related to $W(s)$ by:

$$\Phi(s) = W(s)W'(-s) \tag{1-3}$$

Evidently,

$$\Phi(j\omega) = W(j\omega)W^*(j\omega)$$

so that $\Phi(j\omega)$ is nonnegative hermitian for all ω; when $W(j\omega)$ is a scalar, so is $\Phi(j\omega)$ with $\Phi(j\omega) = |W(j\omega)|^2$.

In the matrix case, Φ is singular for some ω only if W does not have full rank there, and in the scalar case only if W has a zero there.

Spectral factorization, as shown in Example 1-1, seeks a $W(j\omega)$, given $\Phi(j\omega)$. In the rational case, a $W(j\omega)$ exists if and only if $\Phi(j\omega)$ is

nonnegative hermitian for all ω. If Φ is scalar, then $\Phi(j\omega)\geq 0$ for all ω. Normally one restricts attention to $\Phi(\cdot)$ with $\lim_{\omega\to\infty}\Phi(j\omega)<\infty$. A key result is that, given a rational, nonnegative hermitian $\Phi(j\omega)$ with $\lim_{\omega\to\infty}\Phi(j\omega)<\infty$, there exists a rational $W(s)$ where,

- $W(\infty)<\infty$.
- $W(s)$ is stable.
- $W(s)$ is minimum phase, that is, the rank of $W(s)$ is constant in $Re[s]>0$.

In the scalar case, all zeros of $W(s)$ lie in $Re[s]\leq 0$, or in $Re[s]<0$ if $\Phi(j\omega)>0$ for all ω.

In the matrix case, and if $\Phi(j\omega)$ is nonsingular for some ω, it means that $W(s)$ is square and $W^{-1}(s)$ has all its poles in $Re[s]\leq 0$, or in $Re[s]<0$ if $\Phi(j\omega)$ is nonsingular for all ω.

Moreover, the particular $W(s)$ previously defined is unique, to within right multiplication by a constant orthogonal matrix. In the scalar case, this means that $W(s)$ is determined to within a ±1 multiplier.

Example 1-1 Example of Spectral Factorization

Suppose:

$$\Phi(j\omega) = \frac{\omega^2 + 1}{\omega^2 + 4}$$

Then Equation 1-3 is satisfied by $W(s) = \pm\frac{s+1}{s+2}$, which is stable and minimum phase.

Also, Equation 1-3 is satisfied by $\frac{s-1}{s+2}$ and $\frac{s-3}{s+2}$, $\frac{s-1}{s+2}$ and $e^{-sT}\frac{s+1}{s+2}$, and so forth, but none of these is minimum phase.

`bst( )` and `mulhank( )` both require execution within the program of a spectral factorization; the actual algorithm for achieving the spectral factorization depends on a Riccati equation. The concepts of a spectrum and spectral factor also underpin aspects of `wtbalance( )`.

Low Order Controller Design Through Order Reduction

The Model Reduction Module is particularly suitable for achieving low order controller design for a high order plant. This section explains some of the broad issues involved.

Most modern controller design methods, for example, *LQG* and H_∞, yield controllers of order roughly comparable with that of the plant. It follows that, to obtain a low order controller using such methods, one must either follow a high order controller design by a controller reduction step, or reduce an initially given high order plant model, and then design a controller using the resulting low order plant, with the understanding that the controller will actually be used on the high order plant. Refer to Figure 1-2.

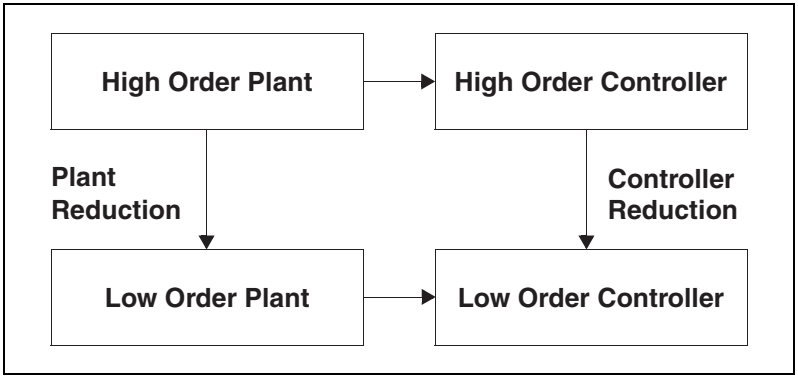

Figure 1-2. Low Order Controller Design for a High Order Plant

Generally speaking, in any design procedure, it is better to postpone approximation to a late step of the procedure: if approximation is done early, the subsequent steps of the design procedure may have unpredictable effects on the approximation errors. Hence, the scheme based on high order controller design followed by reduction is generally to be preferred.

Controller reduction should aim to preserve closed-loop properties as far as possible. Hence the controller reduction procedures advocated in this module reflect the plant in some way. This leads to the frequency weighted reduction schemes of `wtbalance( )` and `fracred( )`, as described in Chapter 4, *Frequency-Weighted Error Reduction*. Plant reduction logically should also seek to preserve closed-loop properties, and thus should involve the controller. With the controller unknown however, this is impossible. Nevertheless, it can be argued, on the basis of the high loop gain property within the closed-loop bandwidth that is typical of many systems, that

multiplicative reduction, as described in Chapter 4, *Frequency-Weighted Error Reduction*, is a sound approach. Chapter 3, *Multiplicative Error Reduction*, and Chapter 4, *Frequency-Weighted Error Reduction*, develop these arguments more fully.

2

Additive Error Reduction

This chapter describes additive error reduction including discussions of truncation of, reduction by, and perturbation of balanced realizations.

Introduction

Additive error reduction focuses on errors of the form,

$$\left\| G(j\omega) - G_r(j\omega) \right\|_\infty$$

where G is the originally given transfer function, or model, and G_r is the reduced one. Of course, in discrete-time, one works instead with:

$$\left\| G(e^{j\omega}) - G_r(e^{j\omega}) \right\|_\infty$$

As is argued in later chapters, if one is reducing a plant that will sit inside a closed loop, or if one is reducing a controller, that again is sitting in a closed loop, focus on additive error model reduction may not be appropriate. It is, however, extremely appropriate in considering reducing the transfer function of a filter. One pertinent application comes specifically from digital filtering: a great many design algorithms lead to a finite impulse response (FIR) filter which can have a very large number of coefficients when poles are close to the unit circle. Model reduction provides a means to replace an FIR design by a much lower order infinite impulse response (IIR) design, with close matching of the transfer function at all frequencies.

Truncation of Balanced Realizations

A group of functions can be used to achieve a reduction through truncation of a balanced realization. This means that if the original system is

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u$$

$$y = \begin{bmatrix} C_1 & C_2 \end{bmatrix} x + D_u \tag{2-1}$$

and the realization is internally balanced, then a truncation is provided by

$$\dot{x}_1 = A_{11} x_1 + B_1 u$$
$$y = C_1 x_1 + Du$$

The functions in question are:

- `balmoore( )`
- `balance( )` (refer to the *Xmath Help*)
- `truncate( )`
- `redschur( )`

One only can speak of internally balanced realizations for systems which are stable; if the aim is to reduce a transfer function matrix $G(s)$ which contains unstable poles, one must additively decompose it into a stable part and unstable part, reduce the stable part, and then add the unstable part back in. The function `stable( )`, described in Chapter 5, *Utilities*, can be used to decompose $G(s)$. Thus:

$G(s) = G_s(s) + G_u(s)$ ($G_s(s)$ stable, $G_u(s)$ unstable)

$G_{sr}(s)$ = found by algorithm (reduction of $G_s(s)$)

$G_r(s) = G_{sr}(s) + G_u(s)$ (reduction of $G(s)$)

A very attractive feature of the truncation procedure is the availability of an error bound. More precisely, suppose that the controllability and observability grammians for [Enn84] are

$$P = Q = \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \tag{2-2}$$

with the diagonal entries of Σ in decreasing order, that is, $\sigma_1 \geq \sigma_2 \geq \cdots$. Then the key result is,

$$\left\| G(j\omega) - G_r(j\omega) \right\|_\infty \leq 2 tr \Sigma_2$$

with G, G_r the transfer function matrices of Equation 2-1 and Equation 2-2, respectively. This formula shows that small singular values can, without great cost, be thrown away. It also is valid in discrete time, and can be improved upon if there are repeated Hankel singular values. Provided that the smallest diagonal entry of Σ_1 strictly exceeds the largest diagonal entry of Σ_2, the reduced order system is guaranteed to be stable.

Several other points concerning the error can be made:

- The error $G(j\omega)–G_r(j\omega)$ as a function of frequency is not flat; it is zero at $\omega = \infty$, and may take its largest value at $\omega = 0$, so that there is in general no matching of DC gains of the original and reduced system.
- The actual error *may* be considerably less than the error bound at all frequencies, so that the error bound formula can be no more than an advance guide. However, the bound is tight when the dimension reduction is 1 and the reduction is of a continuous-time transfer-function matrix.
- With $g(\cdot)$ and $g_r(\cdot)$ denoting the impulse responses for impulse responses of G and G_r and with G_r of degree k, the following L_1 bound holds [GCP88]

  $$\left\| g - g_r \right\|_1 \leq (4\sqrt{2k+1}) tr \Sigma_2$$

  This bound also will apply for the L_∞ error on the step response.

It is helpful to note one situation where reduction is likely to be difficult (so that Σ will contain few diagonal entries which are, relatively, very small). Suppose $G(s)$, strictly proper, has degree n and has $(n – 1)$ unstable zeros. Then as ω runs from zero to infinity, the phase of $G(s)$ will change by $(2n – 1)\pi/2$. Much of this change may occur in the passband. Suppose $G_r(s)$ has degree $n–1$; it can have no more than $(n – 2)$ zeros, since it is strictly

proper. So, even if all zeros are unstable, the maximum phase shift when ω moves from 0 to ∞ is $(2n - 3)\pi/2$. It follows that if $G(j\omega)$ remains large in magnitude at frequencies when the phase shift has moved past $(2n - 3)\pi/2$, approximation of G by G_r will necessarily be poor. Put another way, good approximation may depend somehow on removing roughly cancelling pole-zeros pairs; when there are no left half plane zeros, there can be no rough cancellation, and so approximation is unsatisfactory.

As a working rule of thumb, if there are p right half plane zeros in the passband of a strictly proper $G(s)$, reduction to a $G_r(s)$ of order less than $p + 1$ is likely to involve substantial errors. For non-strictly proper $G(s)$, having p right half plane zeros means that reduction to a $G_r(s)$ of order less than p is likely to involve substantial errors.

An all-pass function exemplifies the problem: there are n stable poles and n unstable zeros. Since all singular values are 1, the error bound formula indicates for a reduction to order $n - 1$ (when it is not just a bound, but exact) a maximum error of 2.

Another situation where poor approximation can arise is when a highly oscillatory system is to be replaced by a system with a real pole.

Reduction Through Balanced Realization Truncation

This section briefly describes functions that `reduce( )`, `balance( )`, and `truncate( )` to achieve reduction.

- `balmoore( )`—Computes an internally balanced realization of a system and optionally truncates the realization to form an approximation.
- `balance( )`—Computes an internally balanced realization of a system.
- `truncate( )`—This function truncates a system. It allows examination of a sequence of different reduced order models formed from the one balanced realization.
- `redschur( )`—These functions in theory function almost the same as the two features of `balmoore( )`. That is, they produce a state-variable realization of a reduced order model, such that the transfer function matrix of the model could have resulted by truncating a balanced realization of the original full order transfer function matrix. However, the initially given realization of the original transfer function matrix is never actually balanced, which can be a numerically hazardous step. Moreover, the state-variable realization of the reduced

order model is not one in general obtainable by truncation of an internally-balanced realization of the full order model.

Figure 2-1 sets out several routes to a reduced-order realization. In continuous time, a truncation of a balanced realization is again balanced. This is *not* the case for discrete time, but otherwise it looks the same.

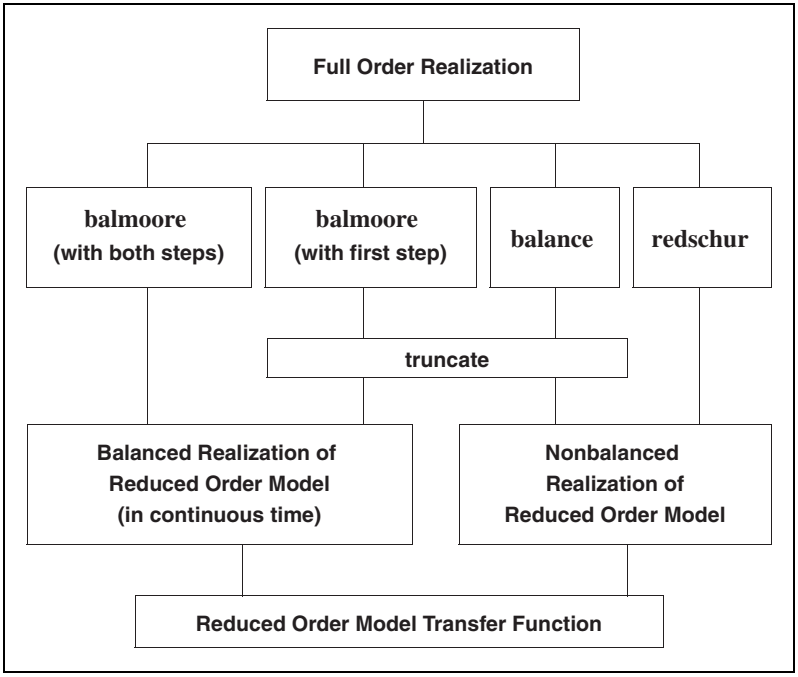

Figure 2-1. Different Approaches for Obtaining the Same Reduced Order Model

Singular Perturbation of Balanced Realization

Singular perturbation of a balanced realization is an attractive way to produce a reduced order model. Suppose $G(s)$ is defined by,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u$$

$$y = \begin{bmatrix} C_1 & C_2 \end{bmatrix} x + Du$$

with controllability and observability grammians given by,

$$P = Q = \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}$$

in which the diagonal entries of Σ are in decreasing order, that is, $\sigma_1 \geq \sigma_2 \geq \cdots$, and such that the last diagonal entry of Σ_1 exceeds the first diagonal entry of Σ_2. It turns out that $Re\lambda_i(A_{22}^{-1})<0$ and $Re\lambda_i(A_{11}-A_{12}A_{22}^{-1}A_{21})< 0$, and a reduced order model $G_r(s)$ can be defined by:

$$\dot{x} = (A_{11} - A_{12}A_{22}^{-1}A_{21})x + (B_1 + -A_{12}A_{22}^{-1}B_2)u$$

$$y = (C_1 - C_2A_{22}^{-1}A_{21})x + (D - C_2A_{22}^{-1}B_2)u$$

The attractive feature [LiA89] is that the same error bound holds as for balanced truncation. For example,

$$\left\| G(j\omega) - G_r(j\omega) \right\|_\infty \leq 2tr\Sigma_2$$

Although the error bounds are the same, the actual frequency pattern of the errors, and the actual maximum modulus, need not be the same for reduction to the same order. One crucial difference is that balanced truncation provides exact matching at $\omega = \infty$, but does not match at DC, while singular perturbation is exactly the other way round. Perfect matching at DC can be a substantial advantage, especially if input signals are known to be band-limited.

Singular perturbation can be achieved with `mreduce( )`. Figure 2-1 shows the two alternative approaches. For both continuous-time and discrete-time reductions, the end result is a balanced realization.

Hankel Norm Approximation

In Hankel norm approximation, one relies on the fact that if one chooses an approximation to exactly minimize one norm (the Hankel norm) then the infinity norm will be approximately minimized. The Hankel norm is defined in the following way. Let $G(s)$ be a (rational) stable transfer

function matrix. Consider the way the associated impulse response maps inputs defined over $(-\infty,0]$ in L_2 into outputs, and focus on the output over $[0,\infty)$. Define the input as $u(t)$ for $t < 0$, and set $v(t) = u(-t)$. Define the output as $y(t)$ for $t > 0$. Then the mapping is

$$y(t) = \int_0^\infty C\exp A(t+r)Bv(r)dr$$

if $G(s) = C(sI\text{-}A)^{-1}B$. The norm of the associated operator is the Hankel norm $\|G\|_H$ of G. A key result is that if $\sigma_1 \geq \sigma_2 \geq \cdots$, are the Hankel singular values of $G(s)$, then $\|G\|_H = \sigma_1$.

To avoid minor confusion, suppose that all Hankel singular values of G are distinct. Then consider approximating G by some stable $\hat{G}$ of prescribed degree k much that $\|G - \hat{G}\|_H$ is minimized. It turns out that

$$\inf_{\hat{G} \text{ of degree k}} \|G - \hat{G}\|_H = \sigma_{k+1}(G)$$

and there is an algorithm available for obtaining $\hat{G}$. Further, the optimum $\hat{G}$ which is minimizing $\|G - \hat{G}\|_H$ does a reasonable job of minimizing $\|G - \hat{G}\|_\infty$, because it can be shown that

$$\|G - \hat{G}\|_\infty \leq \sum_{j=k+1} \sigma_j$$

where $n = \deg G$, with this bound subject to the proviso that G and $\hat{G}$ are allowed to be nonzero and different at $s = \infty$.

The bound on $\|G - \hat{G}\|$ is one half that applying for balanced truncation. However,

- It is actual error that is important in practice (not bounds).
- The Hankel norm approximation does not give zero error at $\omega = \infty$ or at $\omega = 0$. Balanced realization truncation gives zero error at $\omega = \infty$, and singular perturbation of a balanced realization gives zero error at $\omega = 0$.

There is one further connection between optimum Hankel norm approximation and L_∞ error. If one seeks to approximate G by a sum $\hat{G} + F$, with $\hat{G}$ stable and of degree k and with F unstable, then:

$$\inf_{\hat{G} \text{ of degree k and F unstable}} \|G - \hat{G} - F\|_\infty = \sigma_{k+1}(G)$$

Further, the $\hat{G}$ which is optimal for Hankel norm approximation also is optimal for this second type of approximation.

In Xmath Hankel norm approximation is achieved with `ophank( )`. The most comprehensive reference is [Glo84].

balmoore( )

```
[SysR,HSV,T] = balmoore(Sys,{nsr,bound})
```

The `balmoore( )` function computes an internally-balanced realization of a continuous system and then optionally truncates it to provide a balance reduced order system using B.C. Moore's algorithm.

When `balmoore( )` is being used to *reduce* a system, its objective mirrors that of `redschur( )`, therefore, if the same `Sys` and `nsr` are used for both algorithms, the reduced order system should have the same transfer function (though in general the state-variable realizations will be different).

When `balmoore( )` is being used to *balance* a system, its objective, like that of balance, is to generate an internally-balanced state-variable realization. The implementations are not identical.

`balmoore( )` only can be applied on systems that have a stable A matrix, and are controllable and observable, (that is, minimal). Checks, which are rather time-consuming, are included. The computation is intrinsically not well-conditioned if `Sys` is nearly nonminimal. The first part of `balmoore( )` serves to find a transformation matrix T such that the controllability and observability grammians after transformation are equal, and diagonal, with decreasing entries down the diagonal, that is, the system representation is internally balanced. (The condition number of T is a measure of the ill-conditioning of the algorithm. If there is a problem with ill-conditioning, `redschur( )` can be used as an alternative.) If this common grammian is Σ, then after transformation:

(continuous) $\Sigma A' + A\Sigma = -BB' \quad \Sigma A + A'\Sigma = -C'C$

(discrete) $\Sigma - A\Sigma A = -BB' \quad \Sigma - A'\Sigma A = -C'C$

$\Sigma = \text{diag}[\sigma_1, \sigma_2, \sigma_3..., \sigma_{ns}]$ with $\sigma_i \leq \sigma_{i+1} > 0$ with the σ_i the Hankel Singular Values of `Sys`. In the second part of `balmoore( )`, a truncation

of the balanced system occurs, (assuming `nsr` is less than the number of states). Thus, if the state-space representation of the balanced system is

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \qquad C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$$

with A_{11} possessing dimension `nsr` × `nsr`, B_1 possessing `nsr` rows and C_1 possessing `nsr` columns, the reduced order system `SysR` is:

(continuous)	(discrete)
$\dot{x}_1 = A_{11}x_1 + B_1u$	$x_1(k+1) = A_{11}x_1(k) + B_1u(k)$
$y = C_1x + Du$	$y(k) = C_1x_1(k) + Du(k)$

The following error formula is relevant:

(continuous)

$$\left\| [C(j\omega I - A)^{-1}] - [C_1(j\omega I - A_1)^{-1}(B_1 + D)] \right\|_\infty \leq 2[\sigma_{nsr+1} + \sigma_{nsr+2} + \ldots + \sigma_{ns}]$$

(discrete)

$$\left\| [C(e^{j\omega} I - A)^{-1} B + D] - [C_1(e^{j\omega} I - A_1)^{-1} B_1 + D] \right\|_\infty \leq 2[\sigma_{nsr+1} + \sigma_{nsr+2} + \ldots + \sigma_{ns}]$$

It is this error bound which is the basis of the determination of the order of the reduced system when the keyword `bound` is specified. If the error bound sought is smaller than $2\sigma_{ns}$, then no reduction is possible which is consistent with the error bound. If it is larger than $2tr\Sigma$, then the constant transfer function matrix D achieves the bound.

For continuous systems, the actual approximation error depends on frequency, but is always zero at $\omega = \infty$. In practice it is often greatest at $\omega = 0$; if the reduction of state dimension is 1, the error bound is exact, with the maximum error occurring at DC. The bound also is exact in the special case of a single-input, single-output transfer function which has poles and zero alternating along the negative real axis. It is far from exact when the poles and zeros approximately alternate along the imaginary axis (with the poles stable).

The actual approximation error for discrete systems also depends on frequency, and can be large at $\omega = 0$. The error bound is almost never tight, that is, the actual error magnitude as a function of ω almost never attains the error bound, so that the bound can only be a guide to the selection of the reduced system dimension.

In principle, the error bound formula for both continuous and discrete systems can be improved (that is, made tighter or less likely to overestimate the actual maximum error magnitude) when singular values occur with multiplicity greater than one. However, because of errors arising in calculation, it is safer to proceed conservatively (that is, work with the error bound above) when using the error bound to select `nsr`, and examine the actual error achieved. If this is smaller than required, a smaller dimension for the reduced order system can be selected.

`mreduce( )` provides an alternative reduction procedure for a balanced realization which achieves the same error bound, but which has zero error at $\omega = 0$. For continuous systems there is generally some error at $\omega = \infty$, because the D matrix is normally changed. (This means that normally the approximation of a strictly proper system through `mreduce( )` will not be strictly proper, in contrast to the situation with `balmoore( )`.) For discrete systems the D matrix is also normally changed so that, for example, a system which was strictly causal, or guaranteed to contain a delay (that is, $D = 0$), will be approximated by a system `SysR` without this property.

The presentation of the Hankel singular values may suggest a logical dimension for the reduced order system; thus if $\sigma_k \gg \sigma_{k+1}$, it may be sensible to choose $nsr = k$.

With `mreduce( )` and a continuous system, the reduced order system `SysR` is internally balanced, with the grammian $\mathrm{diag}[\sigma_1, \sigma_2, ..., \sigma_{nsr}]$, so that its Hankel Singular Values are a subset of those of the original system `Sys`. Provided $\sigma_{nsr} > \sigma_{nsr+1}$, `SysR` also is controllable, observable, and stable. This is not guaranteed if $\sigma_{nsr} = \sigma_{nsr+1}$, so it is highly advisable to avoid this situation. Refer to the *balmoore( )* section for more on the `balmoore( )` algorithm.

With `mreduce( )` and discrete systems, the reduced order system `SysR` is not in general balanced (in contrast to `balmoore( )`), and its Hankel singular values are not in general a subset of those of `Sys`. Provided $\sigma_{nsr} > \sigma_{srn+1}$, the reduced order system `SysR` also is controllable, observable and stable. This is not guaranteed if $\sigma_{nsr} = \sigma_{nsr+1}$, so it is highly advisable to avoid this situation. For additional information about the `balmoore( )` function, refer to the *Xmath Help*.

Related Functions

balance(), truncate(), redschur(), mreduce()

truncate( )

```
SysR = truncate(Sys,nsr,{VD,VA})
```

The truncate( ) function reduces a system Sys by retaining the first nsr states and throwing away the rest to form a system SysR.

If for Sys one has,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \qquad C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$$

the reduced order system (in both continuous-time and discrete-time cases) is defined by A_{11}, B_1, C_1, and D. If Sys is balanced, then SysR is an approximation of Sys achieving a certain error bound. truncate( ) may well be used after an initial application of balmoore( ) to further reduce a system should a larger approximation error be tolerable. Alternatively, it may be used after an initial application of balance( ) or redschur( ). If Sys was calculated from redschur( ) and VA,VD were posed as arguments, then SysR is calculated as in redschur( ) (refer to the *redschur( )* section).

truncate( ) should be contrasted with mreduce( ), which achieves a reduction through a singular perturbation calculation. If Sys is balanced, the same error bound formulas apply (though not necessarily the same errors), truncate( ) always ensures exact matching at s = ∞ (in the continuous-time case), or exacting matching of the first impulse response coefficient D (in the discrete-time case), while mreduce( ) ensures matching of DC gains for Sys and SysR in both the continuous-time and discrete-time case. For a additional information about the truncate( ) function, refer to the *Xmath Help*.

Related Functions

balance(), balmoore(), redschur(), mreduce()

redschur( )

```
[SysR,HSV,slbig,srbig,VD,VA] = redschur(Sys,{nsr,bound})
```

The `redschur( )` function uses a Schur method (from Safonov and Chiang) to calculate a reduced version of a continuous or discrete system without balancing.

Algorithm

The objective of `redschur( )` is the same as that of `balmoore( )` when the latter is being used to reduce a system; this means that if the same `Sys` and `nsr` are used for both algorithms, the reduced order system should have the same transfer function matrix. However, in contrast to `balmoore( )`, `redschur( )` do not initially transform `Sys` to an internally balanced realization and then truncate it; nor is `SysR` in balanced form. The fact that there is no balancing offers numerical advantages, especially if `Sys` is nearly nonminimal.

`Sys` should be stable, and this is checked by the algorithm. In contrast to `balmoore( )`, minimality of `Sys` (that is, controllability and observability) is not required.

If the Hankel singular values of `Sys` are ordered as $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{ns} \geq 0$, then those of `SysR` in the continuous-time case are $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{nsr} > 0$. A restriction of the algorithm is that $\sigma_{nsr} > \sigma_{nsr+1}$ is required for both continuous-time and discrete-time cases. Under this restriction, `SysR` is guaranteed to be stable and minimal.

The algorithms depend on the same algorithm, apart from the calculation of the controllability and observability grammians W_c and W_o of the original system. These are obtained as follows:

(continuous) $W_c A' + A W_c = -BB'$ $W_o A + A' W_o = C'C$

(discrete) $W_c - A W_c A' = BB'$ $W_o - A' W_o A = C'C$

The maximum order permitted is the number of nonzero eigenvalues of $W_c W_o$ that are larger than ε.

Next, Schur decompositions of W_cW_o are formed with the eigenvalues of W_cW_o in ascending and descending order. These eigenvalues are the square of the Hankel singular values of `Sys`, and if `Sys` is nonminimal, some can be zero.

$$V'_A W_c W_o V_A = S_{asc}$$

$$V'_D W_c W_o V_D = S_{des}$$

The matrices V_A, V_D are orthogonal and S_{asc}, S_{des} are upper triangular. Next, submatrices are obtained as follows:

$$V_{lbig} = V_A \begin{bmatrix} 0 \\ I_{nsr} \end{bmatrix} \qquad V_{rbig} = V_D \begin{bmatrix} I_{nsr} \\ 0 \end{bmatrix}$$

and then a singular value decomposition is found:

$$U_{ebig} S_{ebig} V_{ebig} = V'_{lbig} V_{rbig}$$

From these quantities, the transformation matrices used for calculating `SysR` are defined:

$$S_{lbig} = V_{lbig} U_{ebig} S_{ebig}^{1/2}$$

$$S_{rbig} = V_{rbig} V_{ebig} S_{ebig}^{1/2}$$

and the reduced order system is:

$$A_R = S'_{lbig} A S_{rbig} \qquad B_R = S'_{lbig} B$$

$$A_R = C S_{rbig} \qquad D$$

An *error bound* is available. In the continuous-time case it is as follows. Let $G(j\omega)$ and $G_R(j\omega)$ be the transfer function matrices of `Sys` and `SysR`, respectively.

For the continuous case:

$$\left\| G(j\omega) - G_R(j\omega) \right\|_\infty \leq 2(\sigma_{nsr+1} + \sigma_{nsr+2} + \ldots + \sigma_{ns})$$

For the discrete-time case:

$$\left\| G(e^{j\omega}) - G_R(e^{j\omega}) \right\|_\infty \leq 2(\sigma_{nsr+1} + \sigma_{nsr+2} + \ldots + \sigma_{ns})$$

When `{bound}` is specified, the error bound just enunciated is used to choose the number of states in `SysR` so that the bound is satisfied and `nsr` is as small as possible. If the desired error bound is smaller than $2\sigma_{ns}$, no reduction is made.

In the continuous-time case, the error depends on frequency, but is always zero at $\omega = \infty$. If the reduction in dimension is 1, or the system `Sys` is single-input, single-output, with alternating poles and zeros on the real axis, the bound is tight. It is far from tight when the poles and zeros approximately alternate along the $j\omega$-axis. It is not normally tight in the discrete-time case, and for both continuous-time and discrete-time cases, it is not tight if there are repeated singular values.

The presentation of the Hankel singular values may suggest a logical dimension for the reduced order system; thus if $\sigma_k \gg \sigma_{k+1}$, it may be sensible to choose $nsr = k$.

Related Functions

`ophank()`, `balmoore()`

ophank( )

```
[SysR,SysU,HSV] = ophank(Sys,{nsr,onepass})
```

The `ophank( )` function calculates an optimal Hankel norm reduction of `Sys`.

Restriction

This function has the following restriction:

- Only continuous systems are accepted; for discrete systems use `makecontinuous( )` before calling `bst( )`, then discretize the result.

  ```
  Sys=ophank(makecontinuous(SysD));
  SysD=discretize(Sys);
  ```

Algorithm

The algorithm does the following. The system `Sys` and the reduced order system `SysR` are stable; the system `SysU` has all its poles in $Re[s] > 0$. If the transfer function matrices are $G(s)$, $G_r(s)$ and $G_u(s)$ then:

- $G_r(s)$ is a stable approximation of $G(s)$.
- $G_r(s) + G_u(s)$ is a more accurate, but not stable, approximation of $G(s)$, and optimal in a certain sense.

Of course, the algorithm works with state-space descriptions; that of $G(s)$ can be minimal, while that of $G_r(s)$ cannot be.

These statements are explained in the *Behaviors* section. If `onepass` is specified, reduction is calculated in one pass. If `onepass` is not called or is set to 0 `{onepass=0}`, reduction is calculated in (number of states of `Sys - nsr`) passes. There seems to be no general rule to suggest which setting produces the more accurate approximation G_r. Therefore, if accuracy of approximation for a given order is critical, both should be tried. As noted previously, if an approximation involving an unstable system is desired, the default `{onepass=1}` is specified.

Behaviors

The following explanation deals first with the keyword `{onepass}`. Suppose that $\sigma_1, \sigma_2, ..., \sigma_{ns}$ are the Hankel Singular values of S, which has transfer function matrix $G(s)$. Suppose that the singular values are ordered so that:

$$\sigma_1 = \sigma_2 = \ldots = \sigma_{n_1} > \sigma_{n_1+1} \cdots = \sigma_{n_1+1} \cdots = \sigma_{n_2} > \sigma_{n_2+1} \cdots$$
$$> \sigma_{n_{m-1}+1} = \sigma_{n_{m-1}+2} = \sigma_{n_m} (= \sigma_{ns}) \geq 0$$

Thus, there are n_1 equal values, followed by $n_2 - n_1$ equal values, followed by $n_3 - n_2$ equal values, and so forth.

The order `nsr` of $G_r(s)$ cannot be arbitrary when there are equal Hankel singular values. In fact, the orders shown in Table 2-1 for the strictly stable G_r (all poles in $Re[s]<0$) and strictly unstable G_u (all poles $Re[s]>0$) are possible (and there are no other possibilities).

Table 2-1. Orders of G

Order of G_r nsr	Order of G_u nsu	Number of Eliminated States (Retaining G_u)	Number of Eliminated States (Discarding G_u)
0	$ns - n_1$	n_1	ns
n_1	$ns - n_2$	$n_2 - n_1$	$ns - n_1$
n_2	$ns - n_3$	$n_3 - n_2$	$ns - n_2$
⇓	⇓	⇓	⇓
n_{m-1}	0	$ns - n_{m-1}$	$ns - n_{m-1}$

By abuse of notation, when we say that G is reduced to a certain order, this corresponds to the order of $G_r(s)$ alone; the unstable part of $G_u(s)$ of the approximation is most frequently thrown away. The number of eliminated states (retaining G_u) refers to:

$$(\#\text{ of states in } G) - (\#\text{ of states in } G_r) - (\#\text{ of states in } G_u)$$

This number is always the multiplicity of a Hankel singular value. Thus, when the order of G_r is n_{i-1} the number of eliminated states is $n_i - n_{i-1}$ or the multiplicity of $\sigma_{n_{i-1}+1} = \sigma_{ni}$.

For each order n_{i-1} of $G_r(s)$, it is possible to find G_r and G_u so that:

$$\left\| G(j\omega) - G_r(j\omega) - G_u(j\omega) \right\|_\infty \le \sigma_{n_i}$$

(Choosing $i = 1$ causes G_r to be of order zero; identify $n_0 = 0$.) Actually, among all "approximations" of $G(s)$ with stable part restricted to having degree n_{i-1} and with no restriction on the degree of the unstable part, one can never obtain a lower bound on the approximation error than σ_{n_i}; in the scalar or SISO $G(s)$ case, the $G_r(s)$ which achieves the previous bound is unique, while in the matrix or MIMO $G(s)$ case, the $G_r(s)$ which achieves the previous bound may not be unique [Glo84]. The algorithm we use to find $G_r(s)$ and $G_u(s)$ however allows no user choice, and delivers a single pair of transfer function matrices.

The transfer function matrix $G_r(j\omega)$ alone can be regarded as a stable approximation of $G(j\omega)$. If the D matrix in $G_r(j\omega)$ is approximately chosen, (and the algorithm ensures that it is), then:

$$\left\| G(j\omega) - G_r(j\omega) \right\|_\infty \le \sigma_{n_i} + \sigma_{n_{i+1}} + \ldots + \sigma_{ns} \qquad (2\text{-}3)$$

Thus, the penalty for not being allowed to include G_u in the approximation is an increase in the error bound, by $\sigma_{n_i+1} + ... + \sigma_{ns}$. A number of theoretical developments hinge on bounding the Hankel singular values of $G_r(s)$ and $G_u(-s)$ in terms of those of $G(s)$. With $G_r(s)$ of order n_{i-1}, there holds:

$$\sigma_k(G_r) \leq \sigma_k(G) k = 1, 2, \ldots, n_{i-1}$$

The transfer function matrix $G_u(s)$, being unstable, does not have Hankel singular values; however, $G_u(-s)$ (which is stable) does have Hankel singular values. They satisfy:

$$\sigma_k[G_u(-s)] \leq \sigma_{n_i+k}(G)$$

In most cases, the Hankel singular values of $G(s)$ are distinct. If, accordingly,

$$\left\| G - Gr - G_u \right\|_\infty = \sigma_i$$

then G_r has degree $(i-1)$, G_u has degree $ns-i$ and

$$\left\| G - G_r \right\|_\infty = \sigma_i + \sigma_{i+1} + ... + \sigma_{ns} \quad (2\text{-}4)$$

Observe that the bound (Equation 2-3 or Equation 2-4), which is not necessarily obtained, is one half that applying for both balanced truncation (as implemented by `balmoore( )` or, effectively, by `redschur( )`); it also is one half that obtained when applying `mreduce` to a balanced realization. In general, the D matrices of G and G_r are different, that is, $G(\infty) \neq G_r(\infty)$ (in contrast to `balmoore( )` and `redschur( )`). Similarly, $G(0) \neq G_r(0)$ in general (in contrast to the result when `mreduce` is applied to a balanced realization). The price paid for obtaining a smaller error bound overall through Hankel norm reduction is that one no longer (normally) secures zero error at $\omega = \infty$ or $\omega = 0$.

Two special cases should be noted. If `nsr` = 0 then $G_r(s)$ is a constant only. This constant can be added onto $G_u(s)$, so that $G(s)$ is then being approximated by a totally unstable transfer function matrix, with error σ_1; this type of approximation is known as Nehari approximation. The second special case arises when `nsr` = n_{m-1} (or $NS-1$ if the smallest Hankel singular value has multiplicity 1). In this case, $G_u(s)$ becomes a constant, which can then be lumped in with $G_r(s)$, so that $G(s)$, of degree NS, is then

being approximated by a stable $G_r(s)$ with the actual error (as opposed to just the error bound) satisfying:

$$\left\| G(s) - G_r(s) \right\|_\infty = \sigma_{ns}$$

Note G_r is optimal, that is, there is no other G_r achieving a lower bound.

Onepass Algorithm

The first steps of the algorithm are to obtain the Hankel singular values of $G(s)$ (by using `hankelsv( )`) and identify their multiplicities. (Stability of $G(s)$ is checked in this process.) If the user has specified `nsr` and this does not coincide with one of $0, n_1, n_2, \ldots$ an error message is obtained; generally, all the σ_i are different, so the occurrence of error messages will be rare. The next step of the algorithm is to calculate the sum $G(s) = G_r(s) + G_u(s)$, following [SCL90]. (A separate function `ophred( )` is called for this purpose.) The controllability and observability grammians P and Q are found in the usual way.

$$AP + PA' = -BB'$$
$$QA + A'Q = -C'C$$

and then a singular value decomposition is obtained of the matrix $QP - \sigma_{n_i}^2 I$:

$$\begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_B & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1' \\ V_2' \end{bmatrix} = QP - \sigma_{n_i}^2 I$$

There are precisely $n_i - n_{i-1}$ zero singular values, this being the multiplicity of σ_{n_i}. Next, the following definitions are made:

$$\begin{bmatrix} \overline{A}_{11} & \overline{A}_{12} \\ \overline{A}_{21} & \overline{A}_{22} \end{bmatrix} = \begin{bmatrix} U_1' \\ U_2' \end{bmatrix} (\sigma_{n_i}^2 A' + QAP)(V_1 V_2)$$

$$\begin{bmatrix} \overline{B}_1 \\ \overline{B}_2 \end{bmatrix} = \begin{bmatrix} U_1' \\ U_2' \end{bmatrix} QB$$

$$[\overline{C}_1 \quad \overline{C}_2] = CP[V_1 \quad V_2]$$

and finally:

$$\tilde{A} = S_B^{-1}(\bar{A}_{11} - \bar{A}_{-12}\bar{A}_{22}^{\#}\bar{A}_{21})$$
$$\tilde{B} = S_B^{-1}(\bar{B}_1 - \bar{A}_{-12}\bar{A}_{22}^{\#}\bar{B}_2)$$
$$\tilde{C} = \bar{C}_1 - \bar{C}_2\bar{A}_{22}^{\#}\bar{A}_{21}^{\#}$$
$$\tilde{D} = \bar{D} - C_2\bar{A}_{22}^{\#}\bar{B}_2$$

These four matrices are the constituents of the system matrix of $\tilde{G}(s)$, where:

$$\tilde{G}(s) = G_r(s) + G_u(s)$$

Digression:

This choice is related to the ideas of [Glo84] in the following way; in [Glo84], the complete set is identified of $\tilde{G}(s)$ satisfying

$$\left\| G(j\omega) - \tilde{G}(j\omega) \right\|_\infty = \sigma_{n_i}$$

with $\tilde{G}$ having a stable part of order n_{i-1}. The set is parameterized in terms of a stable transfer function matrix $K(s)$ which has to satisfy

$$C_2 + K(s)B_2^{'} = 0$$
$$I - K'(-j\omega)K(j\omega) \le 0 \text{ for all } \omega$$

with C_2, B_2 being two matrices appearing in the course of the algorithm of [Glo84], and enjoying the property $C_2^{'}C_2 = B_2B_2^{'}$. The particular choice

$$K(s) = -C_2(C'_2C_2)^{\#}B_2$$

in the algorithm of [Glo84] and flagged in corollary 7.3 of [Glo84] is equivalent to the previous construction, in the sense of yielding the same $\tilde{G}_s$, though the actual formulas used here and in [Glo84] for the construction procedure are quite different. In a number of situations, including the case of scalar (*SISO*)$G(s)$, this is the only choice.

The next step of the algorithm is to call `stable( )`, to separate $\tilde{G}(s)$ into its stable and unstable parts, call them $\tilde{G}(s)$ and $\tilde{G}_u(s)$, `stable( )` will always assign the matrix $\tilde{D}$ to $\tilde{G}_r(s)$, and the final step of the algorithm is

to choose the D matrix of $G_r(s)$, by splitting $\tilde{D}$ between $G_r(s)$ and $G_u(s)$. This is done by using a separate function `ophiter( )`. Suppose $G_u(s)$ is the unstable output of `stable( )`, and let $K(s) = G_u(-s)$. By applying the multipass Hankel reduction algorithm, described further below, $K(s)$ is reduced to the constant K_0 (the approximation), which satisfies,

$$\begin{aligned}\left\|K(s) - K_0\right\|_\infty &\le \sigma_1(K) + \ldots + \sigma\ \sigma_{n_s - n_i}(K) \\ &\le \sigma_{n_i + 1}(G) + \ldots + \sigma_{n_s}(G)\end{aligned}$$

that is, if it is larger than,

$$\left\|G_u(-s) - K_0\right\|_\infty \le \sum_{k = n_i + 1}^{n_s} \sigma_k(G)$$

then one chooses:

$$\begin{aligned}G_r &= \tilde{G}_r + K_0 \\ G_u &= \tilde{G}_u + K_0\end{aligned}$$

This ensures satisfaction of the error bound for $G - G_r$ given previously, because:

$$\begin{aligned}\left\|G - G_r\right\|_\infty &= \left\|G - \tilde{G}_r - \tilde{G}_u + (\tilde{G}_u - K_0)\right\|_\infty \\ &= \left\|G - \tilde{G}_r - \tilde{G}_u\right\|_\infty + \left\|K - K_0\right\|_\infty \\ &\le \sigma_{n_i}(G) + \sigma_{n_i + 1}(G) + \ldots + \sigma_{n_s}(G)\end{aligned}$$

Multipass Algorithm

We now explain the multipass algorithm. For simplicity in first explaining the idea, suppose that the Hankel singular values at every stage or pass are distinct.

1. Find a stable order $ns - 1$ approximation $G_{n-1}(s)$ of $G(s)$ with:

$$\left\|G(j\omega) - G_{ns-1}j\omega\right\|_\infty = \sigma_{ns}(G)$$

(This can be achieved by the algorithm already given, and there is no unstable part of the approximation.)

2. Find a stable order $ns - 2$ approximation G_{ns-2} of $G_{ns-1}(s)$, with

$$\left\| G_{ns-1}(j\omega) - G_{ns-2}(j\omega) \right\|_\infty = \sigma_{ns-1}(G_{ns-1})$$

$$\vdots$$

3. (Step *ns–nr)*:

   Find a stable order *nsr* approximation of G_{nsr+1},
   with $\left\| G_{nsr+1}(j\omega) - G_{nsr}(j\omega) \right\|_\infty = \sigma_{nsr+1}(G_{nsr+1})$
   Then, because $\sigma_i(G_{ns-1}) \le \sigma_i(G)$ for $i < ns$,

$$\sigma_i(G_{ns-2}) \le \sigma_i(G_{ns-1})$$

   for $i \le s - i$, ..., this being a property of the algorithm, there follows:

$$\left\| G(j\omega) - G_{nsr}(j\omega) \right\| \le \sigma_{nsr+1}(G_{nsr+1}) + \ldots + \sigma_{ns}(G)$$

$$\le \sum_{i = nsr+1}^{ns} \sigma_i(G)$$

The only difference that arises when singular values have multiplicity in excess of 1 is that the degree reduction at a given step is greater. Thus, if $\sigma_{ns}(G)$ has multiplicity k, then $G(s)$ is approximated by $G_{ns-k}(s)$ of degree $ns - k$.

No separate optimization of the D matrix of G_{nsr} is required. The approximation error bound is the same as for the first algorithm. The actual approximation error may be different. Notice that this second algorithm does not calculate an unstable $G_u(s)$ such that

$$\left\| G(j\omega) - G_{nsr}(j\omega) - G_u(j\omega) \right\|_\infty = \sigma_{nsr+1}$$

Discrete-Time Systems

No special algorithm is presented for discrete-time systems. Any stable discrete-time transfer-function matrix $G(z)$ can be used to define a stable continuous-time transfer-function matrix by a bilinear transformation, thus

$$H(s) = G\left(\frac{\alpha + s}{\alpha - s}\right)$$

when α is a positive constant.

We use *sysZ* to denote G(z) and define:

```
bilinsys=makepoly([-1,a]/makepoly([1,a])
```

as the mapping from the z-domain to the s-domain. The specification is reversed because this function uses backward polynomial rotation. Hankel norm reduction is then applied to H(s), to generate, a stable reduced order approximation $H_r(s)$ and unstable $H_u(s)$ such that:

$$\|H - H_r - H_u\| = \sigma_i$$

$$\|H - H_r\| = \sigma_i + \sigma_{n_i+1} + \ldots + \sigma_{ns}$$

Here, the s_{ni} are the Hankel singular values of both G(z) and H(s); they are the same:

$$G_r(z) = H_r\left(\alpha\frac{z-1}{z+1}\right)$$

$$G_u(z) = H_u\left(\alpha\frac{z-1}{z+1}\right)$$

We then implement the s-domain equivalent with:

```
sysS=subsys(sysZ,bilinsys)
```

There is no simple rule for choosing α; the choice $\alpha = 1$ is probably as good as any. The orders of G_r and G_u are the same as those of H_r and H_u, respectively. The error formulas are as follows:

$$\left\|G(e^{j\omega}) - G_r(e^{j\omega}) - G_u(e^{j\omega})\right\|_\infty = \sigma_{n_i}$$

$$\left\|G(e^{j\omega}) - G_r(e^{j\omega})\right\|_\infty \le \sigma_{n_i} + \sigma_{n_i+1} + \ldots\sigma_{ns}$$

Impulse Response Error

If G_r is determined by the first (single-pass) algorithm, the impulse response error (for $t > 0$) between the impulse responses of G and G_r can be bounded. As shown in Corollary 9.9 of [Glo84], if G_r is of degree $i - 1$ and the multiplicity of the ith larger singular value σ_i of G is r, then:

$$\sigma_j[G - G_r] \le \sigma_i G \text{ for } j = 1, 2, \ldots, 2i - 2 + r$$

$$\le \sigma_{j-i+1}(G) \text{ for } j = 2i - 1 + r, \ldots, ns + i - 1$$

It follows by a result of [BoD87] that the impulse response error for $t > 0$ satisfies:

$$\left\| g(t) - g_r(t) \right\| 1 \leq 2 \left[(2i - 2 + r)\sigma_i(G) + \sum_{i+r}^{ns} \sigma_j(G) \right]$$

Evidently, Hankel norm approximation ensures some form of approximation of the impulse response too.

Unstable System Approximation

A transfer function $G(s)$ with stable and unstable poles can be reduced by applying `stable( )` to separate $G(s)$ into a stable and unstable part. The former is reduced and then the unstable part can be added back on. For additional information about the `ophank( )` function, refer to the *Xmath Help*.

Related Functions

`stable()`, `redschur()`, `balmoore()`

3

Multiplicative Error Reduction

This chapter describes multiplicative error reduction presenting two reasons to consider multiplicative rather than additive error reduction, one general and one specific.

Selecting Multiplicative Error Reduction

The general reason to use multiplicative error reduction is that many specifications are given using decibels; ±1 db corresponds to a multiplicative error of about 12%. Specifications regarding phase shift also can be regarded as multiplicative error statements: ±0.05 radians of phase shift is like 5% multiplicative error also.

The more specific reason arises in considering the problem of plant approximation, with a high order (possibly very high order) plant being initially prescribed, with no controller having been designed, and with a requirement to provide a simpler model of the plant, possibly to allow controller design. Consider the arrangement of Figure 3-1, controller $C(s)$ designed for $G(s)j$ with $G(s) = (I + \Delta)G(s)$.

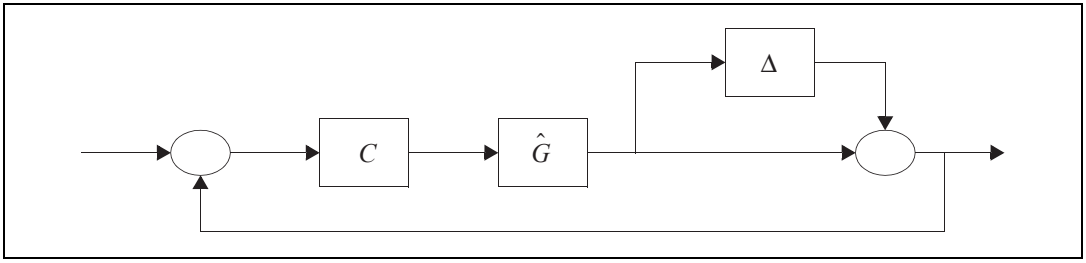

Figure 3-1. Controller $C(s)$ Designed for Multiplicative Error Reduction

The full order plant is $G = (I + \Delta)G$, and the reduced order model is $\hat{G}$. Since $(G - \hat{G})\hat{G}^{-1} = \Delta$, this means that Δ is the multiplicative error. Another way one could measure the multiplicative error would be as $(G - \hat{G})\hat{G}^{-1}$. In the matrix plant case, interchange of the order of the product gives two more possibilities again.

The following multiplicative robustness result can be found in [Vid85].

Multiplicative Robustness Result

Suppose C stabilizes $\hat{G}$, that $\Delta = (G - \hat{G})\hat{G}^{-1}$ has no $j\omega$-axis poles, and that G has the same number of poles in $Re[s] \geq 0$ as $\hat{G}$. If for all ω,

$$\|\Delta(j\omega)\| \left\| \hat{G}(j\omega)C(j\omega)[I + \hat{G}(j\omega)C(j\omega)]^{-1} \right\| < 1 \qquad (3\text{-}1)$$

then C stabilizes G.

This result indicates that if a controller C is designed to stabilize a nominal or reduced order model $\hat{G}$, satisfaction of Equation 3-1 ensures that the controller also will stabilize the *true* plant G.

In reducing a model of the plant, there will be concern not just to have this type of stability property, but also concern to have as little error as possible between the designed system (based on $\hat{G}$) and the true system (based on G). Extrapolation of the stability result then suggests that the goal should be not just to have Equation 3-1, but to minimize the quantity on the left side of Equation 3-1, or its greatest value:

$$\max_{\omega} \{ \|\Delta(j\omega)\| \left\| \hat{G}(j\omega)C(j\omega)[I + \hat{G}(j\omega)C(j\omega)]^{-1} \right\| \}$$

However, there are difficulties. The principal one is that if we are reducing the plant without knowledge of the controller, we cannot calculate the measure because we do not know $C(j\omega)$. Nevertheless, one could presume that, for a well designed system, $\hat{G}C(I + \hat{G}C)^{-1}$ will be close to I over the operating bandwidth of the system, and have smaller norm than 1 (tending to zero as $\omega \to \infty$ in fact) outside the operating bandwidth of the system. This suggests that in the absence of knowledge of C, one should carry out multiplicative approximation by seeking to minimize:

$$\max_{\omega} \|\Delta(j\omega)\| = \|\Delta(j\omega)\|_{\infty}$$

This is the prime rationale for (unweighted) multiplicative reduction of a plant.

Two other points should be noted. First, because frequencies well beyond the closed-loop bandwidth, $\hat{G}C(I + \hat{G}C)^{-1}$ will be small, it is in a sense, wasteful to seek to have $\Delta(j\omega)$ small at very high frequencies. The choice of $\max_{\omega}\|\Delta(j\omega)\|$ as the index is convenient, because it removes a requirement to make assumptions about the controller, but at the same time it does not allow $\|\Delta(j\omega)\|$ to be made even smaller in the closed-loop

bandwidth at the expense of being larger outside this bandwidth, which would be preferable.

Second, the previously used multiplicative error is $(G - \hat{G})\hat{G}^{-1}$. In the algorithms that follow, the error $\delta = (G - \hat{G})\hat{G}^{-1}$ appears. It is easy to check that:

$$\|\delta(j\omega)\|_\infty \leq \frac{\|\Delta(j\omega)\|_\infty}{1 - \|\Delta(j\omega)\|_\infty}$$

and

$$\|\Delta(j\omega)\|_\infty \leq \frac{\|\delta(j\omega)\|_\infty}{1 - \|\delta(j\omega)\|_\infty}$$

This means that if either bound is small, so is the other, with the bounds approximately equal.

Two algorithms for multiplicative reduction are presented: `bst( )`, a mnemonic for balanced stochastic truncation, and `mulhank( )`. Roughly, they relate to one another in the same way that `redschur( )` and `ophank( )` relate, that is, one focuses on balanced realization truncation and the other on Hankel norm approximation. Some of the similarities and differences are as follows:

- When the errors are small, the error bound formula for `bst( )` is about one half of that for `bst( )`.
- With `bst( )`, the actual multiplicative error as a function of frequency goes to zero as $\omega \to \infty$ (or, after using an optional transformation given in the algorithm description, to zero as $\omega \to 0$); with `mulhank( )`, the error tends to be more uniform as a function of frequency.
- `bst( )` can handle nonsquare reduction, while `mulhank( )` cannot.
- Both algorithms are restricted to stable $G(s)$; both preserve right half plane zeros, that is, these zeros are copied into the reduced order object; both have difficulties with $j\omega$-axis zeros of $G(s)$, but these difficulties are not insuperable.

bst( )

```
[SysR,HSV] = bst(Sys,{nsr,left,right,bound,method})
```

The `bst( )` function calculates a balanced stochastic truncation of `Sys` for the multiplicative case.

The objective of the algorithm is to approximate a high-order stable transfer function matrix $G(s)$ by a lower-order $G_r(s)$ with either `inv(g)(g-gr)` or `(g-gr)inv(g)` minimized, under the condition that G_r is stable and of the prescribed order.

Restrictions

This function has the following restrictions:

- The user must ensure that the input system is stable and nonsingular at s = infinity.
- The algorithm may be problematic if the input system has a zero on the $j\omega$-axis.
- Only continuous systems are accepted; for discrete systems use `makecontinuous( )` before calling `bst( )`, then discretize the result.

```
Sys=bst(makecontinuous(SysD));
SysD=discretize(Sys);
```

Algorithm

The modifications described in this section allow you to circumvent the previous restrictions.

The objective of the algorithm is to approximate a high order stable transfer function matrix $G(s)$ by a lower order $G_r(s)$ with, in the square $G(s)$ case, either $\left\|(G-G_r)G^{-1}\right\|_\infty$ or $\left\|G^{-1}(G-G_r)\right\|_\infty$ (approximately) minimized, under the constraint that G_r is stable and of prescribed order `nsr`. In case G is not square but has full row rank, the algorithm seeks to minimize:

$$\left\|(G-G_r)_*(GG_*)^{-1}(G-G_r)\right\|_\infty$$

Recall that $X_*(s) = X'(-s)$ so that when $s = j\omega$,

$$X_*(j\omega) = X^*(j\omega)$$

When G is not square but has full column rank, the algorithm seeks to minimize:

$$\left\|(G-G_r)(G_*G)^{-1}(G-G_r)_*\right\|_\infty$$

These cases are secured with the keywords `right` and `left`, respectively. If the wrong option is requested for a nonsquare $G(s)$, an error message will result.

The algorithm has the property that right half plane zeros of $G(s)$ remain as right half plane zeros of $G_r(s)$. This means that if $G(s)$ has order nsr with n_+ zeros in $Re[s] > 0$, $G_r(s)$ must have degree at least n_+, else, given that it has n_+ zeros in $Re[s] > 0$ it would not be proper, [Gre88].

The conceptual basis of the algorithm can best be grasped by considering the case of scalar $G(s)$ of degree n. Then one can form a minimum phase, stable $W(s)$ with $|W(j\omega)|^2 = |G(j\omega)|^2$ and then an all-pass function (the *phase function*) $W^{-1}(-s)\,G(s)$. This all pass function has a mixture of stable and unstable poles, and it encodes the phase of $G(j\omega)$. Its stable part has n Hankel singular values σ_i with $\sigma_i \leq 1$, and the number of σ_i equal to 1 is the same as the number of zeros of $G(s)$ in $Re[s] > 0$. State-variable realizations of W,G and the stable part of $W^{-1}(-s)G(s)$ can be connected in a nice way, and when the stable part of $W^{-1}(-s)G(s)$ has a balanced realization, we say that the realization of G is stochastically balanced. Truncating the balanced realization of the stable part of $W^{-1}(-s)G(s)$ induces a corresponding truncation in the realization of $G(s)$, and the truncated realization defines an approximation of G. Further, a good approximation of a transfer function encoding the phase of G somehow ensures a good approximation, albeit in a multiplicative sense, of G itself.

Algorithm with the Keywords right and left

The following description of the algorithm with the keyword `right` is based on ideas of [GrA86] developed in [SaC88]. The procedure is almost the same when `left` is specified, except the transpose of $G(s)$ is used; the algorithm finds an approximation in the same manner as for `right`, but transposes the approximation to yield the desired $G_r(s)$.

1. The algorithm checks
   - That the system is state-space, continuous, and stable
   - That a correct option has been specified if the plant is nonsquare
   - That D is nonsingular; if the plant is nonsquare, DD´ must be nonsingular

2. With $G(s) = D + C(sI - A)^{-1}B$ and stable, with DD' nonsingular and $G(j\omega)\ G'(-j\omega)$ nonsingular for all ω, part of a state variable realization of a minimum phase stable $W(s)$ is determined such that $W'(-s)W(s) = G(s)G'(-s)$ with

$$W(s) = D_W + C_W(sI - A_w)^{-1}B_W$$

The state variable matrices in $W(s)$ are obtained as follows. The controllability grammian P associated with $G(s)$ is first found from $AP + PA' + BB'=0$ then $A_W = A$, $B_W = PC' + BD'$.

When $G(s)$ is square, the algorithm checks to see if there is a zero or singularity of $G(s)$ close to the $j\omega$-axis (the zeros are given by the eigenvalues of $A - BD^{-1}C$ and are computed reliably with the aid of `schur( )`). If one is found, you are warned that results may be unreliable. Next, a stabilizing solution Q is found for the following Riccati equation:

$$QA + A'Q + (C - B'_{WQ})'(DD')^{-1}(C - B'_WQ) = 0$$

The `singriccati( )` function is used; failure of the nonsingularity condition on $G(j\omega)G'(-j\omega)$ will normally result in an error message that no stabilizing solution exists. To obtain the best numerical results, `singriccati( )` is invoked with the keyword `{method="schur"}`. Although D_W, C_W are not needed for the remainder of the algorithm, they are simply determined in the square case by

$$D_W = D' \qquad C_W = D^{-1}(C - B'_WQ)$$

with minor modification in the nonsquare case. The real point of the algorithm is to compute P and Q; the matrix Q satisfies (square or nonsquare case).

$$QA + A'Q + C'_WC_W = 0$$

P, Q are the controllability and observability grammians of the transfer function $C_W(sI - A)^{-1}B$. This transfer function matrix, it turns out, is the strictly proper, stable part of $\theta(s) = W^{-T}(-s)G(s)$, which obeys the matrix all-pass property $\theta(s)\theta'(-s) = I$, and is the *phase matrix* associated with $G(s)$.

3. Compute ordered Schur decompositions of PQ, with the eigenvalues of PQ is ascending and descending order. Obtain the phase matrix Hankel singular values, that is, the Hankel singular values of the

strictly proper stable part of $\theta(s)$, as the square roots of the eigenvalues of PQ. Call these quantities ν_i. The Schur decompositions are,

$$V_A' PQ V_A = S_{asc} \qquad V_D' PQ V_D = S_{des}$$

where V_A, V_D are orthogonal and S_{asc}, S_{des} are upper triangular.

4. Define submatrices as follows, assuming the dimension of the reduced order system `nsr` is known:

$$V_{lbig} = V_A \begin{bmatrix} 0 \\ I_{nsr} \end{bmatrix} \qquad V_{rbig} = V_D \begin{bmatrix} I_{nsr} \\ 0 \end{bmatrix}$$

Determine a singular value decomposition,

$$U_{ebig} S_{ebig} V_{ebig} = V_{lbig}' V_{rbig}$$

and then define transformation matrices:

$$S_{lbig} = V_{lbig} U_{ebig} S_{ebig}^{-1/2}$$

$$S_{rbig} = V_{rbig} V_{ebig} S_{ebig}^{-1/2}$$

The reduced order system G_r is:

$$A_R = S_{lbig}' A S_{rbig} \qquad B_R = S_{lbig}' B$$

$$A_R = CS_{rbig} \qquad D_R = D$$

where step 4 is identical with that used in `redschur( )`, except the matrices P, Q which determine V_A, V_D and so forth, are the controllability and observability grammians of $C_W(sI - A)^{-1}B$ rather than of $C(sI - A)^{-1}B$, the controllability grammian of $G(s)$ and the observability grammian of $W(s)$.

The error formula [WaS90] is:

$$\left\| G^{-1}(G - G_r) \right\|_\infty \le 2 \sum \frac{\nu_i}{1 - \nu_i} \tag{3-2}$$

All ν_i obey $\nu_i \le 1$. One can only eliminate ν_i where $\nu_i < 1$. Hence, if `nsr` is chosen so that $\nu_{nsr+1} = 1$, the algorithm produces an error message. The algorithm also checks that `nsr` does not exceed the dimension of a minimal

state-variable representation of G. In this case, the user is effectively asking for $G_r = G$. When the phase matrix has repeated Hankel singular values, they must all be included or all excluded from the model, that is, $\nu_{nsr} = \nu_{nsr+1}$ is not permitted; the algorithm checks for this.

The number of ν_i equal to 1 is the number of zeros in $Re[s]>0$ of $G(s)$, and as mentioned already, these zeros remain as zeros of $G_r(s)$.

If `error` is specified, then the error bound formula (Equation 3-2) in conjunction with the ν_i values from step 3 is used to define `nsr` for step 4. For nonsquare G with more columns than rows, the error formula is:

$$\left\| (G - G_r)_* (G_* G)^{-1} (G - G_r) \right\|_\infty^{1/2} \leq 2 \sum_{i = nsr+1}^{ns} \frac{\nu_i}{1 - \nu_i}$$

If the user is presented with the ν_i, the error formula provides a basis for intelligently choosing `nsr`. However, the error bound is not guaranteed to be tight, except when $nsr = ns - 1$.

Securing Zero Error at DC

The error $G^{-1}(G - G_r)$ as a function of frequency is always zero at $\omega = \infty$. When the algorithm is being used to approximate a high order plant by a low order plant, it may be preferable to secure zero error at $\omega = 0$. A method for doing this is discussed in [GrA90]; for our purposes:

1. We need a bilinear transformation of sys = 1/z. Given $G(s)$ we generate $H(s)$ through:

   ```
   bilinsys=makepoly([b3,b4]/makepoly([b1,b2])
   sys=subsys(sys,bilinsys)
   ```

2. Reduce with the previous algorithm:

   ```
   [sr,nsr,hsv] = bst(sys)
   ```

3. Use the bilinear transformation s = 1/z again:

   ```
   [sr1,nsr1] = bilinear(sr,nsr,[0,1,1,0])
   ```

The ν_i are the same for $G(s)$ and $H(s) = G(s^{-1})$. The error bound formula is the same; H is stable and $H(j\omega)H'(-j\omega)$ of full rank for all ω including $\omega = \infty$ if and only if G has the same property; right half plane zeros of G are still preserved by the algorithm. The error $G^{-1}(G - G_r)$, though now zero at $\omega = 0$, is in general nonzero at $\omega = \infty$.

Hankel Singular Values of Phase Matrix of G_r

The ν_i, $i = 1,2,...,ns$ have been termed above the Hankel singular values of the phase matrix associated with G. The corresponding quantities for G_r are ν_i, $i = 1,..., nsr$.

Further Error Bounds

The introduction to this chapter emphasized the importance of the error measures

$$\left\|(G - G_r)G_r^{-1}\right\|_\infty \text{ or } \left\|G_r^{-1}(G - G_r)\right\|$$

for plant reduction, as opposed to $\left\|(G - G_r)G^{-1}\right\|_\infty$ or $\left\|G^{-1}(G - G_r)\right\|_\infty$

The BST algorithm ensures that in addition to (Equation 3-2), there holds [WaS90a].

$$\left\|G_r^{-1}(G - G_r)\right\|_\infty \le 2 \sum_{i = nsr + 1}^{ns} \frac{\nu_i}{1 - \nu_i}$$

which also means that for a scalar system,

$$\left|20\log_{10}\frac{G_r}{G}\right| \le 8.69\left(2 \sum_{i = nsr + 1}^{ns} \frac{\nu_i}{1 - \nu_i}\right) dB$$

and, if the bound is small:

$$\left|\text{phase}(G) - \text{phase}(G_r)\right| \le \sum_{i = nsr + 1}^{ns} \frac{\nu_i}{1 - \nu_i} \; radians$$

Reduction of Minimum Phase, Unstable G

For square minimum phase but not necessarily stable G, it also is possible to use this algorithm (with minor modification) to try to minimize (for G_r of a certain order) the error bound

$$\left\|(G - G_r)G_r^{-1}\right\|_\infty$$

which also can be relevant in finding a reduced order model of a plant. The procedure requires G again to be nonsingular at $\omega = \infty$, and to have no $j\omega$-axis poles. It is as follows:

1. Form $H = G^{-1}$. If G is described by state-variable matrices A, B, C, D, then H is described by $A - BD^{-1}C$, BD^{-1}, $-D^{-1}C$, D^{-1}. H is square, stable, and of full rank on the $j\omega$-axis.
2. Form H_r of the desired order to minimize approximately:

$$\left\| H^{-1}(H - H_r) \right\|_\infty$$

3. Set $G_r = H^{-1}{}_r$.

   Observe that

$$\begin{aligned} H^{-1}(H - H_r) &= I - H^{-1}H_r \\ &= I - GG_r^{-1} \\ &= (G_r - G)G_r^{-1} \end{aligned}$$

   The reduced order G_r will have the same poles in $Re[s] > 0$ as G, and be minimum phase.

Imaginary Axis Zeros (Including Zeros at ∞)

We shall now explain how to handle the reduction of $G(s)$ which has a rank drop at $s = \infty$ or on the $j\omega$-axis. The key is to use a bilinear transformation, [Saf87]. Consider the bilinear map defined by

$$s = \frac{z - a}{-bz + 1}$$

$$z = \frac{s + a}{bs + 1}$$

where $0 < a < b^{-1}$ and mapping $G(s)$ into $\tilde{G}(s)$ through:

$$\tilde{G}(s) = G\left(\frac{s - a}{-bs + 1}\right)$$

$$G(s) = \tilde{G}\left(\frac{s + a}{bs + 1}\right)$$

The values of $G(s)$, as shown in Figure 3-2, along the $j\omega$-axis are the same as the values of $\tilde{G}(s)$ around a circle with diameter defined by $[a - j0, b^{-1} + j0]$ on the positive real axis.

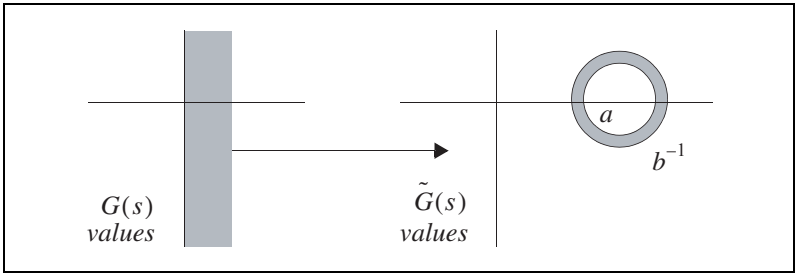

Figure 3-2. Bilinear Mapping from $G(s)$ to $(\tilde{G}s)$ (Case 1)

Also, the values of $\tilde{G}(s)$, as shown in Figure 3-3, along the $j\omega$-axis are the same as the values of $G(s)$ around a circle with diameter defined by $[-b^{-1} + j0, -a + j0]$.

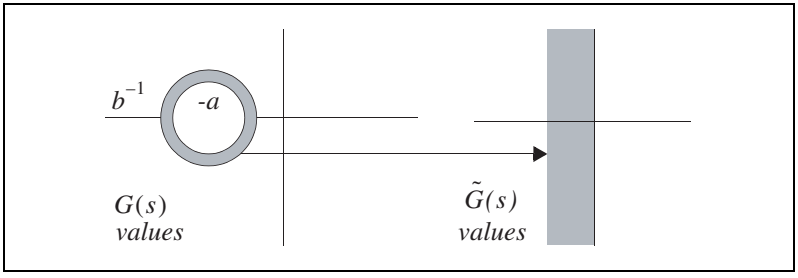

Figure 3-3. Bilinear Mapping from $G(s)$ to $(\tilde{G}s)$ (Case 2)

We can implement an arbitrary bilinear transform using the `subsys( )` function, which substitutes a given transfer function for the s- or z-domain operator.

To implement $\tilde{G}(s) = G\left(\frac{s-a}{-bs+1}\right)$ use:

```
gtildesys=subsys(gsys,makep([-b,1]/makep([1,-a])
```

To implement $G(s) = \tilde{G}\left(\frac{s+a}{s+1}\right)$ use:

```
gsys=subsys(gtildesys,makep([b,1]/makep([1,a])
```

Note The systems substituted in the previous calls to subsys invert the function specification because these functions use backward polynomial rotation.

Any zero (or rank reduction) on the $j\omega$-axis of $G(s)$ becomes a zero (or rank reduction) in $Re[s] > 0$ of $\tilde{G}(s)$, and if $G(s)$ has a zero (or rank reduction) at infinity, this is shifted to a zero (or rank reduction) of $\tilde{G}(s)$ at the point b^{-1}, (in $Re[s] > 0$). If all poles of $G(s)$ are inside the circle of diameter $[-b^{-1} + j0, a + j0]$, all poles of $\tilde{G}(s)$ will be in $Re[s] < 0$, and if $G(s)$ has no zero (or rank reduction) on this circle, $\tilde{G}(s)$ will have no zero (or rank reduction) on the $j\omega$-axis, including $\omega = \infty$.

If $G(s)$ is nonsingular for almost all values of s, it will be nonsingular or have no zero or rank reduction on the circle of diameter $[-b^{-1} + j0, -a + j0]$ for almost all choices of a,b. If a and b are chosen small enough, $G(s)$ will have all its poles inside this circle and no zero or rank reduction on it, while $\tilde{G}(s)$ then will have all poles in $Re[s] < 0$ and no zero or rank reduction on the $j\omega$-axis, including $s = \infty$. The steps of the algorithm, when $G(s)$ has a zero on the $j\omega$-axis or at $s = \infty$, are as follows:

1. For small a,b with $0 < a < b^{-1}$, form $\tilde{G}(s) = G\left(\frac{s-a}{-bs+1}\right)$ as shown for `gtildesys`.
2. Reduce $\tilde{G}(s)$ to $\tilde{G}_r(s)$, this being possible because $\tilde{G}(s)$ is stable and has full rank on $s = j\omega$, including $\omega = \infty$.
3. Form $G_r(s) = \tilde{G}_r\left(\frac{s+a}{bs+1}\right)$ as shown for `gsys`.

The error $\left\|G^{-1}(G - G_r)\right\|_\infty$ will be overbounded by the error $\left\|\tilde{G}^{-1}(\tilde{G} - \tilde{G}_r)\right\|_\infty$, and G_r will contain the same zeros in $Re[s] \geq 0$ as G.

If there is no zero (or rank reduction) of $G(s)$ at the origin, one can take $a = 0$ and b^{-1} = bandwidth over which a good approximation of $G(s)$ is needed, and at the very least b^{-1} sufficiently large that the poles of $G(s)$ lie in the circle of diameter $[-b^{-1} + j0, -a + j0]$. If there is a zero or rank reduction at the origin, one can replace $a = 0$ by $a = b$. It is possible to take b too small, or, if there is a zero at the origin, to take a too small. The user will be presented with an error message that there is a $j\omega$-axis zero and/or that the Riccati equation solution may be in error. The basic explanation is that as $b \to 0$, and thus $a \to 0$, the zeros of $\tilde{G}(s)$ approach those of $G(s)$. Thus, for sufficiently small b, one or more zeros of $\tilde{G}(s)$ may be identified as lying on the imaginary axis. The remedy is to increase a and/or b above the desirable values.

The procedure for handling $j\omega$-axis zeros or zeros at infinity will be deficient if the number of such zeros is the same as the order of $G(s)$—for example, if $G(s) = 1/d(s)$, for some stable $d(s)$. In this case, it is possible

again with a bilinear transformation to secure multiplicative approximations over a limited frequency band. Suppose that

$$\tilde{G}(s) = G\left(\frac{s}{\varepsilon s + 1}\right)$$

Create a system that corresponds to $\tilde{G}(s)$ with:

```
gtildesys=subs(gsys,(makep([-eps,1])/makep([1,-]))
bilinsys=makep([eps,1])/makep([1,0])
sys=subsys(sys,bilinsys)
```

Under this transformation:

- Values of $G(s)$ along the $j\omega$-axis correspond to values of $\tilde{G}(s)$ around a circle in the left half plane on diameter $(-\varepsilon^{-1} + j0, 0)$.
- Values of $\tilde{G}(s)$ along the $j\omega$-axis correspond to values of $G(s)$ around a circle in the right half plane on diameter $(0, \varepsilon^{-1} + j0)$.

Multiplicative approximation of $\tilde{G}(s)$ (along the $j\omega$-axis) corresponds to multiplicative approximation of $G(s)$ around a circle in the right half plane, touching the $j\omega$-axis at the origin. For those points on the $j\omega$-axis near the circle, there will be good multiplicative approximation of $G(j\omega)$. If it is desired to have a good approximation of $G(s)$ over an interval $[-j\Omega, j\Omega]$, then a choice such as $\varepsilon^{-1} = 5\ \Omega$ or $10\ \Omega$ needs to be made. Reduction then proceeds as follows:

1. Form $\tilde{G}(s)$.
2. Reduce $\tilde{G}(s)$ through `bst( )`.
3. Form $G_r(s) = -\tilde{G}_r(s/(1 - \varepsilon s))$ with:

   ```
   gsys=subsys(gtildesys(gtildesys,
   makep([-eps,-1])/makep[-1,-0]))
   ```

Notice that the number of zeros of $G(s)$ in the circle of diameter

$$(0, \varepsilon^{-1} + j0)s$$

sets a lower bound on the degree of $G_r(s)$—for such zeros become right half plane zeros of $\tilde{G}(s)$, and must be preserved by `bst( )`. Obviously, zeros at $s = \infty$ are never in this circle, so a procedure for reducing $G(s) = 1/d(s)$ is available.

There is one potential source of failure of the algorithm. Because $G(s)$ is stable, $\tilde{G}(s)$ certainly will be, as its poles will be in the left half plane circle on diameter $(-\varepsilon = j0, 0)$. If $\tilde{G}_r(s)$ acquires a pole outside this circle (but still in the left half plane of course)—and this appears possible in principle—$G_r(s)$ will then acquire a pole in $Re\,[s] > 0$. Should this difficulty be encountered, a smaller value of ε should be used.

Related Functions

`redschur()`, `mulhank()`

mulhank( )

```
[SysR,HSV] = mulhank(Sys,{nsr,left,right,bound,method})
```

The `mulhank( )` function calculates an optimal Hankel norm reduction of `Sys` for the multiplicative case.

Restrictions

This function has the following restrictions:

- The user must ensure that the input system is stable and nonsingular at s = infinity.
- The algorithm may be problematic if the input system has a zero on the $j\omega$-axis.
- Only continuous systems are accepted; for discrete systems use `makecontinuous( )` before calling `mulhank( )`, then discretize the result.

```
Sys=mulhank(makecontinuous(SysD));
SysD=discretize(Sys);
```

Algorithm

The objective of the algorithm, like `bst( )`, is to approximate a high order square stable transfer function matrix $G(s)$ by a lower order $G_r(s)$ with either $\left\|(G - G_r)G^{-1}\right\|_\infty$ or $\left\|G^{-1}(G - G_r)\right\|_\infty$ (approximately) minimized, under the constraint that G_r is stable and of prescribed order.

The algorithm has the property that right half plane zeros of G(s) are retained as zeros of $G_r(s)$. This means that if G(s) has order NS with N_+ zeros in Re[s] > 0, $G_r(s)$ must have degree at least N_+—else, given that it has N_+ zeros in Re[s] > 0 it would not be proper, [GrA89].

The conceptual basis of the algorithm can best be grasped by considering the case of scalar $G(s)$ of degree n. Then one can form a minimum phase, stable $W(s)$ with $|W(j\omega)|^2 = |G(j\omega)|^2$ and then an all-pass function (the *phase function*) $W^{-1}(-s)\,G(s)$. This all-pass function has a mixture of stable and unstable poles, and it encodes the phase of $G(j\omega)$. Its stable part has n Hankel singular values σ_i with $\sigma_i \le 1$, and the number of σ_i equal to 1 is the same as the number of zeros of $G(s)$ in $Re[s]>0$. State-variable realizations of W,G and the stable part of $W^{-1}(-s)G(s)$ can be connected in a nice way. The algorithm computes an additive Hankel norm reduction of the stable part of $W^{-1}(-s)G(s)$ to cause a degree reduction equal to the multiplicity of the smallest σ_i. The matrices defining the reduced order object are then combined in a new way to define a multiplicative approximation to $G(s)$; as it turns out, there is a close connection between additive reduction of the stable part of $W^{-1}(-s)G(s)$ and multiplicative reduction of $G(s)$. The reduction procedure then can be repeated on the new phase function of the just found approximation to obtain a further reduction again in $G(s)$.

right and left

A description of the algorithm for the keyword `right` follows. It is based on ideas of [Glo86] in part developed in [GrA86] and further developed in [SaC88]. The procedure is almost the same when `{left}` is specified, except the transpose of $G(s)$ is used; the following algorithm finds an approximation, then transposes it to yield the desired $G_r(s)$.

1. The algorithm checks that $G(s)$ is square, stable, and that the transfer function is nonsingular at infinity.
2. With $G(s) = D + C(sI-A)^{-1}B$ square and stable, with D nonsingular [`rank(d)` must equal number of rows in d] and $G(j\omega)$ nonsingular for all finite ω, this step determines a state variable realization of a minimum phase stable $W(s)$ such that,

   $W'(-s)W(s) = G(s)G'(-s)$

   with:

   $W(s) = D_w + C_w(sI-A_w)^{-1}B_w$

   The various state variable matrices in $W(s)$ are obtained as follows. The controllability grammian P associated with $G(s)$ is first found from $AP + PA' + BB' = 0$, then:

   $A_w = AB_w = PC'+BD'D_w = D'$

   The algorithm checks to see if there is a zero or singularity of $G(s)$ close to the $j\omega$-axis. The zeros are determined by calculating the

eigenvalues of A – B/D * C with the aid of `schur( )`. If any real part of the `eigenvalues` is less than `eps`, a warning is displayed.

Next, a stabilizing solution Q is found for the following Riccati equation:

$$QA + A'Q + (C - B'_w Q)'(DD')^{-1}(C - B'_w Q) = 0$$

The function `singriccati( )` is used; failure of the nonsingularity condition of $G(j\omega)$ will normally result in an error message. To obtain the best numerical results, `singriccati( )` is invoked with the keyword `method="schur"`.

The matrix C_w is given by $C_w = D^{-1}(C - B'_w Q)$.

Notice that Q satisfies $QA + A'Q + C'_w C_w = 0$, so that P and Q are the controllability and observability grammians of

$$F(s) = C_w(sI - A)^{-1}B$$

This strictly proper, stable transfer function matrix is the strictly proper, stable part (under additive decomposition) of $\theta(s)=W^{-T}(-s)G(s)$, which obeys the matrix all pass property $\theta(s)\theta'(-s)=I$. It is the *phase matrix* associated with $G(s)$.

3. The Hankel singular values ν_i of $F(s) = C_w(sI - A)^{-1}B$ are computed, by calling `hankelsv( )`. The value of `nsr` is obtained if not prespecified, either by prompting the user or by the error bound formula ([GrA89], [Gre88], [Glo86]).

$$\nu_{nsr+1} \le \left\| G^{-1}(G - G_r) \right\|_\infty \le \prod_{j=nsr+1}^{ns} (1 + \nu_j) - 1 \qquad (3\text{-}3)$$

(with $\nu_i \ge \nu_{i+1} \ge \cdots$ being assumed). If $\nu_k = \nu_{k+1} = \ldots = \nu_{k+r}$ for some k, (that is, ν_k has multiplicity greater than unity), then ν_k appears once only in the previous error bound formula. In other words, the number of terms in the product is equal to the number of distinct ν_i less than ν_{nsr}. There are restrictions on `nsr`. `nsr` cannot exceed the dimension of a minimal realization of $G(s)$; although $\nu_i \ge \nu_{i+1} \cdots$, `nsr` must obey $n_{nsr} > n_{nsr+1}$; and while $1 \ge \nu_i$ for all i, it is necessary that $1 > \nu_{nsr+1}$. (The number of ν_i equal to 1 is the number of right half plane zeros of $G(s)$. They must be retained in $G_r(s)$, so the order of $G_r(s)$, `nsr`, must at least be equal to the number of ν_i equal to 1.) The software checks all these conditions. The minimum order permitted is the number of Hankel

singular values of $F(s)$ larger than $1-\varepsilon$ (refer to steps 1 through 3 of the *Restrictions* section). The maximum order permitted is the number of nonzero eigenvalues of $W_c W_o$ larger than ε.

4. Let r be the multiplicity of ν_{ns}. The algorithm approximates

$$F(s) = C_w(sI - A)^{-1}B$$

by a transfer function matrix $\hat{F}(s)$ of order $ns - r$, using Hankel norm approximation. The procedure is slightly different from that used in `ophank( )`.

Construct an SVD of $QP - \nu_{ns}^2 I$:

$$QP - \nu_{NS}^2 I = U\begin{bmatrix}\Sigma_1 & 0\\ 0 & 0\end{bmatrix} \qquad V' = [U_1 U_2]\begin{bmatrix}\Sigma_1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}V_1'\\ V_2'\end{bmatrix}$$

with Σ_1 of dimension $(ns - r) \times (ns - r)$ and nonsingular. Also, obtain an orthogonal matrix T, satisfying:

$$\overline{B}_2 + \overline{C}'_{w2}T = 0$$

where $\overline{B}_2$ and $\overline{C}'_{w2}$ are the last r rows of $\overline{B}$ and $\overline{C}_w'$, the state variable matrices appearing in a balanced realization of $C'_w s(I - A)^{-1}B$. It is possible to calculate T without evaluating $\overline{B}$ $\overline{B}$, $\overline{C}_w$ as it turns out (refer to [AnJ]), and the algorithm does this. Now with

$$\hat{F}(s) = \hat{D}_F + \hat{C}_F(sI - \hat{A}_F)^{-1}\hat{B}_F$$
$$\hat{F}_p(s) = \hat{C}_F(sI - \hat{A}_F)\hat{B}_F$$

there holds:

$$\hat{A}_F = \Sigma_1^{-1}U_1'[\nu_{ns}^2 A' + QAP - \nu_{ns}C_w'TB']V_1$$
$$\hat{B}_F = \Sigma_1^{-1}U_1'[QB + \nu_{ns}C_w'^T]$$
$$\hat{C}_F = (C_w P + \nu_{ns}TB')V'$$
$$\hat{D}_F = -\nu_{ns}T$$

Note The expression $\hat{F}_p(s)$ is the strictly proper part of $\hat{F}(s)$. The matrix $v_{ns}^{-1}[F(s) - \hat{F}(s)]$ is all pass; this property is not always secured in the multivariable case when `ophank( )` is used to find a Hankel norm approximation of $F(s)$.

5. The algorithm constructs $\hat{G}$ and $\hat{W}$, which satisfy,

$$\hat{G}(s) = G(s) - W'(-s)[F(s) - \hat{F}(s)]$$

and,

$$\hat{W}(s) = (I - v_{ns}T')(I - v_{ns}T)^{-1} \{W(s) - [F(s) - \hat{F}(s)]G' + (-s)\}$$

through the state variable formulas

$$(\hat{G}(s) = (D(I - v_{ns}T))[D\hat{C}_F + B_W'U\Sigma_1](sI - \hat{A}_F)^{-1}\hat{B}_F)$$

and:

$$\hat{W}(s) = (I - v_{ns}T')D' + (I - v_{ns}T')(I - v_{ns}T)^{-1} \hat{C}_F(sI - \hat{A}_F)^{-1}[\hat{B}_F D' + V_1'C']$$

Continue the reduction procedure, starting with $\hat{G}$, $\hat{W}$, $\hat{F}$ and repeating the process till G_r of the desired degree `nsr` is obtained. For example, in the second iteration, $\hat{\hat{G}}(s)$ is given by:

$$\hat{\hat{G}}(s) = \hat{G}(s) - \hat{W}' + (-s)[\hat{F}_p(s) - \hat{\hat{F}}(s)] \quad (3\text{-}4)$$

Consequences of Step 5 and Justification of Step 6

A number of properties are true:

- $\hat{G}(s)$ is of order $ns - r$, with:

$$\left\| G^{-1}(G - \hat{G}) \right\|_\infty = v_{ns} \quad (3\text{-}5)$$

- $\hat{W}(s)$ and $\hat{G}s$ stand in the same relation as $W(s)$ and $G(s)$, that is:
  - $\hat{W}'(-s)\hat{W}(s) = \hat{G}(s)\hat{G}'(-s)$
  - With $\hat{P}\hat{A}'_F + \hat{A}_F\hat{P} = -\hat{B}_F\hat{B}'_F$, there holds

$$B_{\hat{W}} = \hat{P}C'_{\hat{G}} + B_{\hat{G}}D'_{\hat{G}}$$

or

$$\hat{B}_F D' + V_1 C' = \hat{P}(D\hat{C}_F + B'_W U_1\Sigma_1)' + \hat{B}_F(I - v_{ns}T')D'$$

  - With $\hat{Q}\hat{A}_F + \hat{A}'_F\hat{Q} = -\hat{C}'_F\hat{C}_F$ there holds

$$C_{\hat{W}} = D_{\hat{G}}^{-1}(C_{\hat{G}} - B'_{\hat{W}}\hat{Q})$$

or

$$(I - v_{ns}T')(I - v_{ns}T)^{-1}\hat{C}_F = [D(I - v_{ns}T)]^{-1}$$
$$\{D\hat{C}_F + B'_W U_1(\Sigma_1 - [\hat{B}_F D' + V_1 C']'\hat{Q})\}$$

  - $D_{\hat{W}} = D'_{\hat{G}}$
  - $\hat{F}$ is the stable strictly proper part of $(\hat{W}^{-1}(-s))\hat{G}(s)$.
- The Hankel singular values of $\hat{F}_p$(and $\hat{F}$) are the first $as - r$ Hankel singular values of F,

$$\hat{P} = \Sigma_1^{-1}U'_1QV_1 = V'_1QU_1\Sigma_1^{-1}$$
$$\hat{Q} = V'_1PU_1\Sigma_1 = \Sigma_1U'_1PV_1$$

- $\hat{G}s$ has the same zeros in $Re[s] > 0$ as $G(s)$.

These properties mean that one is immediately positioned to repeat the reduction procedure on $\hat{G}s$, with almost all needed quantities being on hand.

Error Bounds

The error bound formula (Equation 3-3) is a simple consequence of iterating (Equation 3-5). To illustrate, suppose there are three reductions $G \to \hat{G} \to \hat{G}_2 \to \hat{G}_3$, each by degree one. Then,

$$\begin{aligned} G^{-1}(G - \hat{G}_3) &= G^{-1}(G - \hat{G}) \\ &\quad + G^{-1}\hat{G}\hat{G}^{-1}(\hat{G} - \hat{G}_2) \\ &\quad + G^{-1}\hat{G}\hat{G}^{-1}\hat{G}_2\hat{G}_2^{-1}(\hat{G}_2 - \hat{G}_3) \end{aligned}$$

Also,

$$\begin{aligned} \left\| G^{-1}\hat{G} \right\| &= \left\| \hat{G}^{-1}(\hat{G} - G) + I \right\| \\ &\le 1 + v_{ns} \end{aligned}$$

Similarly,

$$\left\| \hat{G}^{-1}\hat{G}_2 \right\| \le 1 + v_{ns-1}, \left\| \hat{G}_2^{-1}\hat{G}_3 \right\| \le 1 + v_{ns-2}$$

Then:

$$\begin{aligned} \left\| G^{-1}(G - \hat{G}_3) \right\| &\le v_{ns} + (1 + v_{ns})v_{ns-1} + (1 + v_{ns-1})v_{ns-2} \\ &= (1 + v_{ns})(1 + v_{ns-1})(1 + v_{ns-2}) - 1 \end{aligned}$$

The error bound (Equation 3-3) is only exact when there is a single reduction step. Normally, this algorithm has a lower error bound than `bst( )`; in particular, if the v_i are all distinct and $v_{nsr+1} \ll 1$, the error bounds are approximately

$$\sum_{i=nsr+1}^{ns} v_i \quad \text{for } \texttt{mulhank( )} \qquad\qquad 2\sum_{i=nsr+1}^{ns} v_i \quad \text{for } \texttt{bst(}$$

For `mulhank( )`, this translates for a scalar system into

$$-86.9 \sum_{i=nsr+1}^{ns} v_i \text{ dB} < 20\log_{10}\left|\hat{G}_{nsr}/G\right| < 8.69 \sum_{i=nsr+1}^{ns} v_i \text{ dB}$$

and

$$\left|\text{phase error}\right| < \sum_{i=nsr+1}^{ns} v_i \text{ radians}$$

The bounds are double for `bst( )`.

The error as a function of frequency is always zero at $\omega = \infty$ for `bst( )` (or at $\omega = 0$ if a transformation $s \rightarrow s^{-1}$ is used), whereas no such particular property of the error holds for `mulhank( )`.

Imaginary Axis Zeros (Including Zeros at ∞)

When $G(j\omega)$ is singular (or zero) on the $j\omega$ axis or at ∞, reduction can be handled in the same manner as explained for `bst( )`.

The key is to use a bilinear transformation [Saf87]. Consider the bilinear map defined by

$$s = \frac{z-a}{-bz+1}$$

$$z = \frac{s+a}{bs+1}$$

where $0 < a < b^{-1}$ and mapping $G(s)$ into $\tilde{G}(s)$ through

$$\tilde{G}(s) = G\left(\frac{s-a}{-bs+1}\right)$$

$$G(s) = \tilde{G}\left(\frac{s+a}{bs+1}\right)$$

The values of $G(s)$ along the $j\omega$-axis are the same as the values of $\tilde{G}(s)$ around a circle with diameter defined by $[a - j0, b^{-1} + j0]$ on the positive real axis (refer to Figure 3-2). Also, the values of $\tilde{G}(s)$ along the $j\omega$-axis are the same as the values of $G(s)$ around a circle with diameter defined by $[-b^{-1} + j0, -a + j0]$.

We can implement an arbitrary bilinear transform using the `subsys( )` function, which substitutes a given transfer function for the s- or z-domain operator, as previously shown.

To implement $\tilde{G}(s) = G\left(\frac{s-a}{-bs+1}\right)$ use:

```
gtildesys=subsys(gsys,makep([-b,1]/makep([1,-a])
```

To implement $G(s) = \tilde{G}\left(\frac{s+a}{bs+1}\right)$ use:

```
gsys=subsys(gtildesys,makep([b,1]/makep([1,a])
```

Note The systems substituted in the previous calls to subsys invert the function specification because these functions use backward polynomial rotation.

Any zero (or rank reduction) on the $j\omega$-axis of $G(s)$ becomes a zero (or rank reduction) in $Re[s] > 0$ of $\tilde{G}(s)$, and if $G(s)$ has a zero (or rank reduction) at infinity, this is shifted to a zero (or rank reduction) of $\tilde{G}(s)$ at the point b^{-1}, again in $Re[s] > 0$. If all poles of $G(s)$ are inside the circle of diameter $[-b^{-1} + j0, a + j0]$, all poles of $\tilde{G}(s)$ will be in $Re[s] < 0$, and if $G(s)$ has no zero (or rank reduction) on this circle, $\tilde{G}(s)$ will have no zero (or rank reduction) on the $j\omega$-axis, including $\omega = \infty$.

If $G(s)$ is nonsingular for almost all values of s, it will be nonsingular or have no zero or rank reduction on the circle of diameter $[-b^{-1} + j0, -a + j0]$ for almost all choices of a,b. If a and b are chosen small enough, $G(s)$ will have all its poles inside this circle and no zero or rank reduction on it, while $\tilde{G}(s)$ then will have all poles in $Re[s] < 0$ and no zero or rank reduction on the $j\omega$-axis, including $s = \infty$.

The steps of the algorithm, when $G(s)$ has a zero on the $j\omega$-axis or at $s = \infty$, are as follows:

1. For small a,b with $0 < a < b^{-1}$, form $\tilde{G}(s) = G\left(\frac{s-a}{-bs+1}\right)$ as shown for `gtildesys`.
2. Reduce $\tilde{G}(s)$ to $\tilde{G}_r(s)$, this being possible because $\tilde{G}(s)$ is stable and has full rank on $s = j\omega$, including $\omega = \infty$.
3. Form $G_r(s) = \tilde{G}_r\left(\frac{s+a}{bs+1}\right)$ as shown for `gsys`.

The error $\left\| G^{-1}(G - G_r) \right\|_\infty$ will be overbounded by the error $\left\| \tilde{G}^{-1}(\tilde{G} - \tilde{G}_r) \right\|_\infty$, and G_r will contain the same zeros in $Re[s] \geq 0$ as G.

If there is no zero (or rank reduction) of $G(s)$ at the origin, one can take $a = 0$ and b^{-1} = bandwidth over which a good approximation of $G(s)$ is needed, and at the very least b^{-1} sufficiently large that the poles of $G(s)$ lie in the circle of diameter $[-b^{-1} + j0, -a + j0]$. If there is a zero or rank reduction at the origin, one can replace $a = 0$ by $a = b$. It is possible to take b too small, or, if there is a zero at the origin, to take a too small. In these cases an error message results, saying that there is a $j\omega$-axis zero and/or that the Riccati equation solution may be in error. The basic explanation is that as $b \rightarrow 0$, and thus $a \rightarrow 0$, the zeros of $\tilde{G}(s)$ approach those of $G(s)$. Thus, for sufficiently small b, one or more zeros of $\tilde{G}(s)$ may be identified as lying on the imaginary axis. The remedy is to increase a and/or b above the desirable values.

The previous procedure for handling $j\omega$-axis zeros or zeros at infinity will be deficient if the number of such zeros is the same as the order of $G(s)$; for example, if $G(s) = 1/d(s)$, for some stable $d(s)$. In this case, it is possible again with a bilinear transformation to secure multiplicative approximations over a limited frequency band. Suppose that

$$\tilde{G}(s) = G\left(\frac{s}{\varepsilon s + 1}\right)$$

Create a system that corresponds to $\tilde{G}(s)$ with:

```
gtildesys=subs(gsys,(makep([-eps,1])/makep([1,-]))
bilinsys=makep([eps,1])/makep([1,0])
sys=subsys(sys,bilinsys)
```

Under this transformation:

- Values of $G(s)$ along the $j\omega$-axis correspond to values of $\tilde{G}(s)$ around a circle in the left half plane on diameter $(-\varepsilon^{-1} + j0, 0)$.
- Values of $\tilde{G}(s)$ along the $j\omega$-axis correspond to values of $G(s)$ around a circle in the right half plane on diameter $(0, \varepsilon^{-1} + j0)$.

Multiplicative approximation of $\tilde{G}(s)$ (along the $j\omega$-axis) corresponds to multiplicative approximation of $G(s)$ around a circle in the right half plane, touching the $j\omega$-axis at the origin. For those points on the $j\omega$-axis near the circle, there will be good multiplicative approximation of $G(j\omega)$. If a good approximation of $G(s)$ over an interval $[-j\Omega, j\Omega]$ it is desired, then $\varepsilon^{-1} = 5\ \Omega$ or $10\ \Omega$ are good choices. Reduction then proceeds as follows:

1. Form $\tilde{G}(s)$.
2. Reduce $\tilde{G}(s)$ through `bst( )`.
3. Form $G_r(s) = -\tilde{G}_r(s/(1-\varepsilon s))$ with:

```
gsys=subsys(gtildesys(gtildesys,
makep([-eps,-1])/makep[-1,-0]))
```

Notice that the number of zeros of $G(s)$ in the circle of diameter $(0, \varepsilon^{-1} + j0)$ sets a lower bound on the degree of $G_r(s)$—for such zeros become right half plane zeros of $\tilde{G}(s)$, and must be preserved by `bst( )`. Zeros at $s = \infty$ are never in this circle, so a procedure for reducing $G(s) = 1/d(s)$ is available.

There is one potential source of failure of the algorithm. Because $G(s)$ is stable, $\tilde{G}(s)$ certainly will be, as its poles will be in the left half plane circle on diameter $(-\varepsilon^{-1} = j0, 0)$. If $\tilde{G}_r(s)$ acquires a pole outside this circle (but still in the left half plane of course)—and this appears possible in principle—$G_r(s)$ will then acquire a pole in $Re\,[s] > 0$. Should this difficulty be encountered, a smaller value of ε should be used.

Related Functions

`singriccati(), ophank(), bst(), hankelsv()`

4

Frequency-Weighted Error Reduction

This chapter describes frequency-weighted error reduction problems. This includes a discussion of controller reduction and fractional representations.

Introduction

Frequency-weighted error reduction means that the error is measured not, as previously, by

$$E_0 = \left\| G(j\omega) - G_r(j\omega) \right\|_\infty$$

but rather by

$$E_1 = \left\| G(j\omega) - G_r(j\omega)V(j\omega) \right\|_\infty \tag{4-1}$$

or

$$E_2 = \left\| W(j\omega)[G(j\omega) - G_r(j\omega)] \right\|_\infty \tag{4-2}$$

or

$$E_3 = \left\| W(j\omega)[G(j\omega) - G_r(j\omega)]V(j\omega) \right\|_\infty \tag{4-3}$$

where W,V are certain weighting matrices. Their presence reflects a desire that the approximation process be more accurate at certain frequencies (where V or W have large singular values) than at others (where they have small singular values). For scalar $G(j\omega)$, all the indices above are effectively the same, with the effective weight just $|V(j\omega)|$, $|W(j\omega)|$, or $|W(j\omega)V(j\omega)|$.

When the system G is processing signals which do not have a flat spectrum, and is to be approximated, there is considerable logic in using a weight. If the signal spectrum is $\Phi(j\omega)$, then taking $V(j\omega)$ as a stable spectral factor

(so that $VV^* = \Phi$) is logical. However, a major use of weighting is in controller reduction, which is now described.

Controller Reduction

Frequency weighted error reduction becomes particularly important in reducing controller dimension. The LQG and H_∞ design procedures lead to controllers which have order equal to, or roughly equal to, the order of the plant. Very often, controllers of much lower order will result in acceptable performance, and will be desired on account of their greater simplicity.

It is almost immediately evident that (unweighted) additive approximation of a controller will not necessarily ensure closeness of the behavior of the two closed-loop systems formed from the original and reduced order controller together with the plant. This behavior is dependent in part on the plant, and so one would expect that a procedure for approximating controllers ought in some way to reflect the plant. This can be done several ways as described in the *Controller Robustness Result* section. The following result is a trivial variant of one in [Vid85] dealing with robustness in the face of plant variations.

Controller Robustness Result

Suppose that a controller C stabilizes a plant P, and that C_r is a (reduced order) approximation to C with the same number of unstable poles. Then C_r stabilizes P also provided

$$\left\| [C(j\omega) - C_r(j\omega)]P(j\omega)[I + C(j\omega)P(j\omega)]^{-1} \right\|_\infty < 1$$

or

$$\left\| [I + P(j\omega)C(j\omega)]^{-1}(P(j\omega)[C(j\omega)C_r(j\omega)]) \right\|_\infty < 1$$

An extrapolation to this thinking [AnM89] suggests that C_r will be a good approximation to C (from the viewpoint of some form of stability robustness) if

$$E_{IS} = \left\| (C - C_r)P(I + CP)^{-1} \right\|_\infty$$

or

$$E_{IS} = \left\| (C - C_r)P(I + CP)^{-1} \right\|_\infty$$

is minimized (and of course is less than 1). Notice that these two error measures are like those of Equation 4-1 and Equation 4-2. The fact that the plant ought to show up in a good formulation of a controller reduction problem is evidenced by the appearance of P in the two weights.

It is instructive to consider the shape of the weighting matrix or function $P(I + CP)^{-1}$. Consider the scalar plant case. In the pass band, $|PC|$ is likely to be large, and if this is achieved by having $|C|$ large, then $|P(I + CP)^{-1}|$ will be (approximately) small. Also outside the plant bandwidth, $|P(I + CP)^{-1}|$ will be small. This means that it will be most likely to take its biggest values at frequencies near the unity gain cross-over frequency. This means that the approximation C_r is being forced to be more accurate near this frequency than well away from it—a fact very much in accord with classical control, where one learns the importance of good loop shaping round this frequency.

The above measures E_{IS} and E_{OS} are advanced after a consideration of stability, and the need for its preservation in approximating C by C_r. If one takes the viewpoint that the important thing to preserve is the closed-loop transfer function matrix, a different error measure arises. With T, T_r denoting the closed-loop transfer function matrices,

$$T - T_r = PC(I + PC) - PC_r(I + PC_r)^{-1}$$

Then, to a first order approximation in $C - C_r$, there holds

$$T - T_r \approx (I + PC)^{-1}P(C - C_r)(I + PC)^{-1}$$

The natural error measure is then

$$E_M = \left\|(I + PC)^{-1}P(C - C_r)(I + PC)^{-1}\right\|_\infty \tag{4-4}$$

and this error measure parallels E_3 in Equation 4-3. Further refinement again is possible. It may well be that closed-loop transfer function matrices should be better matched at some frequencies than others; if this weighting on the error in the closed-loop transfer function matrices is determined by the input spectrum $VV^* = \Phi$, then one really wants $(T - T_r)V$ to be small, so that Equation 4-4 is replaced by

$$E_{MS} = \left\|(I + PC)^{-1}P(C - C_r)(I + PC)^{-1}V\right\|_\infty$$

Most of these ideas are discussed in [Enn84], [AnL89], and [AnM89]. The function `wtbalance( )` implements weighted reduction, with five choices of error measure, namely E_{IS}, E_{OS}, E_M, E_{MS}, and E_1 with arbitrary $V(j\omega)$. The first four are specifically for controller reduction, whereas the last is not aimed specifically at this situation.

Several features of the algorithms are:

- Only the stable part of C is really reduced; the unstable part is copied exactly into C_r.
- A modification of balanced realization truncation underpins the algorithms, namely (frequency) weighted balanced truncation, although to avoid numerical problems, the actual construction of a frequency weighted balanced realization of C is avoided.
- Frequency weighted Hankel singular values can be computed, and although no error bound formula is available (in contrast to the unweighted problem), generally speaking there is little damage done in reducing by a number of states equal to the number of (relatively) small Hankel singular values.

The error measures themselves deserve certain comments:

- The two stability based measures E_{IS} and E_{OS} are derived from a sufficiency condition for stability, rather than a necessity and sufficiency condition, and so capture stability a little crudely.
- For any constant nonsingular N, the error measure E_{IS} can be replaced by $\left\| N(C - C_r)P(I + CP)^{-1}N^{-1} \right\|_\infty$ and the robustness result remains valid. Use of an N may improve or worsen the quality of the approximation.
- Having $T - T_r$ small normally ensures closeness of the closed-loop impulse and step responses.
- In classical control especially, constraints on the loop gain can be imposed (Minimum value of gain in one band, maximum value of gain in another band, for example). None of the methods presented directly addresses the task of retaining satisfaction of these constraints after reduction of a high order acceptable controller. However, judicious use of a weight V can assist. Suppose that above the closed-loop bandwidth there is an overbound constraint on the loop gain, which is violated when a controller reduction is performed (but not with the original controller). At these frequencies, roughly PC and PC_r are small, so that $T - T_r = P(C - C_r)$. Introduction of a weight V in E_{MS} penalizing frequencies in the region in question will evidently encourage PC_r to better track PC.

Fractional Representations

The treatment of $j\omega$-axis or right half plane poles in the above schemes is crude: they are simply copied into the reduced order controller. A different approach comes when one uses a so-called matrix fraction description (MFD) to represent the controller, and controller reduction procedures based on these representations (only for continuous-time) are found in `fracred( )`. Consider first a scalar controller $C(s) = n(s)/d(s)$. One can take a stable polynomial $\bar{d}(s)$ of the same degree as d, and then represent the controller as a ratio of two stable transfer functions, namely

$$\left[\frac{n(s)}{\bar{d}(s)}\right] \qquad \left[\frac{d(s)}{\bar{d}(s)}\right]^{-1}$$

Now $n/\bar{d}$ is the numerator, and $d/\bar{d}$ the denominator. Write $d/\bar{d}$ as $1 + e/\bar{d}$. Then we have the equivalence shown in Figure 4-1.

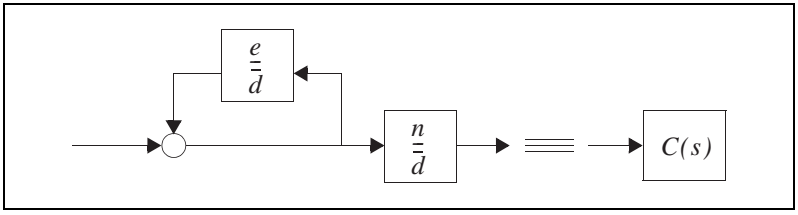

Figure 4-1. Controller Representation Through Stable Fractions

Evidently, $C(s)$ can be formed by completing the following steps:

1. Construction of the one-input, two-output stable transfer function matrix

$$G = \begin{bmatrix} n/\bar{d} \\ e/\bar{d} \end{bmatrix}$$

   (which has order equal to that of $\bar{d}$ or d).

2. Interconnection through negative feedback of the second output to the single input.

   These observations motivate the reduction procedure:

   - Reduce G to G_r; notice that G is stable. Let G_r be

$$G = \begin{bmatrix} n_r/\bar{d}_r \\ e_r/\bar{d}_r \end{bmatrix}$$

- Form the reduced controller by interconnecting using negative feedback the second output of G_r to the input, that is, set

$$C_r(s) = \frac{n_r}{\bar{d}_r + e_r}$$

Nothing has been said as to how $\bar{d}$ should be chosen—and the end result of the reduction, $C_r(s)$, depends on $\bar{d}_r$. Nor has the reduction procedure been specified.

When $C(s)$ has been designed to combine a state estimator with a stabilizing feedback law, it turns out that there is a natural choice for $\bar{d}(s)$.

As for the reduction procedure, one possibility is to use a weight based on the spectrum of the input signals to G—and in case $C(s)$ has been determined by an LQG optimal design, this spectrum turns out to be white, that is, independent of frequency, so that no weight (apart perhaps from scaling) is needed. A second possibility is to use a weight based on a stability robustness measure. These points are now discussed in more detail.

To understand the construction of a natural fractional representation for $C(s)$, suppose that $P(s) = C(sI - A)^{-1}B$ and let K_R, K_E be matrices such that $A - BK_R$ and $A - K_{EC}$ are stable. The controller

$$\begin{aligned}\hat{x} &= A\hat{x} + Bu - K_E(C\hat{x} - y)\\ u &= -K_R\hat{x}\end{aligned}$$

generates an estimate $-K_R\hat{x}$ of the feedback control $-K_R x$. The controller can be represented as a series compensator

$$\begin{aligned}\hat{x} &= A\hat{x} + BK_R\hat{x} - K_E C\hat{x} + K_E y\\ u &= -K_R\hat{x}\end{aligned}$$

(with compensator input y and output u). Allowing for connection with negative feedback, the compensator transfer function matrix is:

$$C(s) = K_R(sI - A + BK_R + K_{EC})^{-1}K_E$$

Matrix algebra shows that $C(s)$ can be described through a left or right matrix fraction description

$$C(s) = D_L^{-1}(s)N_L(s) = N_R(s)D_R^{-1}(s)$$

with D_L, and related values, all stable transfer function matrices. In particular:

$$D_L = I + K_R(sI - A + K_E C)^{-1}B$$
$$N_L = K_R(sI - A + K_E C)^{-1}K_E$$
$$N_R = K_R(sI - A + BK_R)^{-1}K_E$$
$$D_R = I + C(sI - A + BK_R)^{-1}K_E$$

For matrix C(s), the left and right matrix fraction descriptions are distinct entities. It is the right MFD which corresponds to Figure 4-1; refer to Figure 4-2.

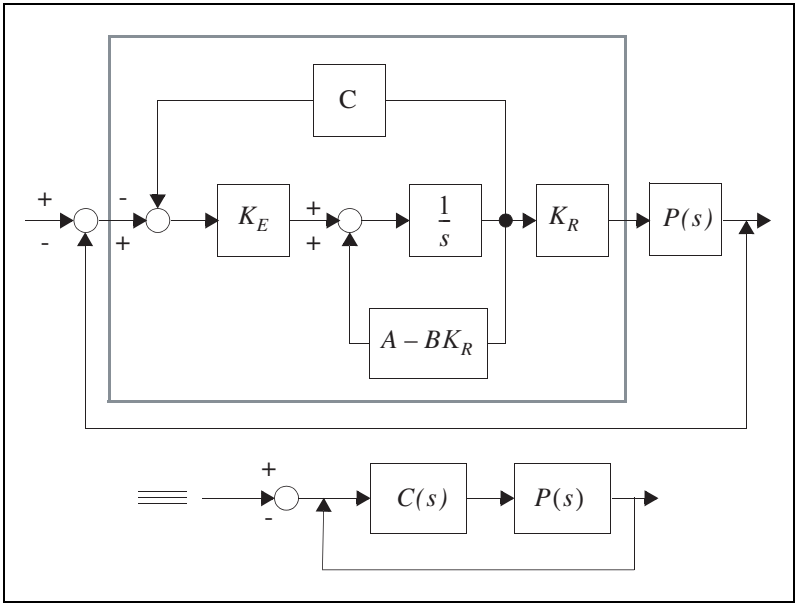

Figure 4-2. C(s) Implemented to Display Right MFD Representation

The left MFD corresponds to the setup of Figure 4-3.

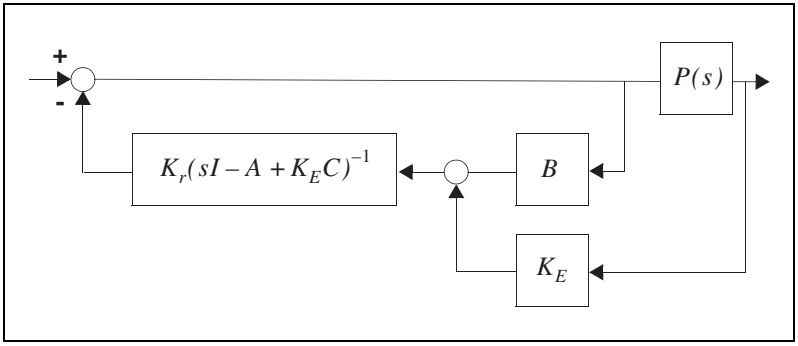

Figure 4-3. C(s) Implemented to Display Left MFD Representation

The setup of Figure 4-2 suggests approximation of:

$$G(s) = \begin{bmatrix} K_r \\ C \end{bmatrix} (sI - A + BK_r)^{-1} K_E$$

whereas that of Figure 4-3 suggests approximation of:

$$H(s) = K_R (sI - A + K_E C)^{-1} \begin{bmatrix} B & K_E \end{bmatrix}$$

In the LQG optimal case, the signal driving K_E in Figure 4-2 is white noise (the innovations process); this motivates the possibility of using no frequency dependent weighting in approximating G(s) [but observe that after approximating, the signal will no longer be white noise, so that argument is questionable]. Simple appeal to duality motivates using no frequency dependent weighting for H(s). These are two of the options offered by `fracred( )`.

Two more `fracred( )` options depend on examining stability robustness (the options are duals of one another). From the stability point of view, the set-up of Figure 4-3 is identical to that of Figure 4-4, with $\hat{P} = \begin{bmatrix} P & I \end{bmatrix}$.

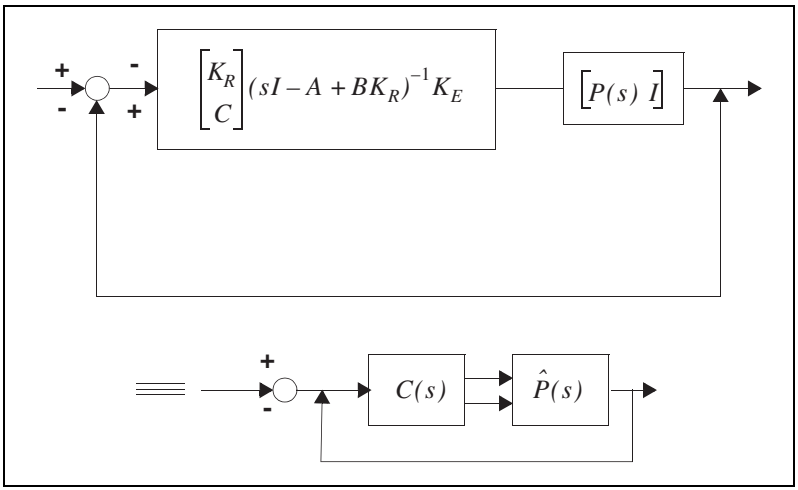

Figure 4-4. Redrawn; Individual Signal Paths as Vector Paths

It is possible to verify that

$$(I + \hat{P}G)^{-1}\hat{P} = [CsI - A + K_E C^{-1}B \quad I - C(sI - (A + K_E C)^{-1}K_E)]$$

and accordingly the output weight $(I + \hat{P}G)^{-1}\hat{P} = W$ can be used in an error measure $\|W(G - G_r)\|$. It turns out that the calculations for frequency weighted balanced truncation of G and subsequent construction of $C_r(s)$ are exceptionally easy using this weight.

The second `fracred( )` option is the dual of this. The error measure is $\|(H - H_r)V\|$ where:

$$V = \begin{bmatrix} I - K_R(sI - A + BK_R)^{-1}B \\ C(sI - A + BK_R)^{-1}B \end{bmatrix}$$

It is possible to argue heuristically the relevance of these error measures from a second point of view. It turns out that:

$$\begin{bmatrix} D_L & N_L \\ -W_1 & W_2 \end{bmatrix}\begin{bmatrix} V_1 & -N_R \\ V_2 & D_r \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

(Here, the W_i and V_i are submatrices of W,V.) Evidently,

$$\begin{bmatrix} D_L & N_L \end{bmatrix} V = I \quad \text{and} \quad W \begin{bmatrix} N_R \\ D_R \end{bmatrix} = I$$

Some manipulation shows that trying to preserve these identities after approximation of D_L, N_L or N_R, D_R suggests use of the error measures $\|W(G-G_r)\|_\infty$ and $\|(H-H_r)V\|_\infty$. For further details, refer to [AnM89] and [LAL90].

In all four `fracred( )` options, it is possible to construct (weighted) Hankel singular values, and to use them as a guide to the likely quality of approximation. The patterns tend to be different for the four options.

The `fracred( )` options are normally different in outcome from the `wtbalance( )` options. However, if the controller has been designed by the loop transfer recovery method and is stable, then one of the `fracred( )` options is essentially the same as one of the `wtbalance( )` options, refer to [LiA90].

More precisely, if the LTR design is performed with input noise or process noise weighting tending to infinity, reduction with `fracred( )` and `type="left stab"`, which uses the error measure $\|(H-H_r)V\|$, leads to effectively the same reduction as `wtbalance ( )` with the `type="input stab"`. If the LTR design is performed with state or output weighting tending to infinity (in the index determining the state feedback law), reduction with `fracred( )` and `type="right stab"` using the error measure $\|W(G-G_r)\|_\infty$ leads to effectively the same reduction as `wtbalance( )` with `type="output stab"`.

wtbalance( )

```
[SysCR,SysCLR,HSV] = wtbalance(Sys,SysC,type,{nscr,SysV})
```

The `wtbalance( )` function calculates a frequency weighted balanced truncation of a system.

`wtbalance( )` has two separate uses:

- Reduce the order of a controller $C(s)$ located in a stable closed-loop, with the plant $P(s)$ known. Frequency-weighted balanced truncation is used, with the weights involving $P(s)$ and being calculated in a predominantly standard way.

- Reduce the order of a transfer function matrix $C(s)$ through frequency-weighted balanced truncation, a stable frequency weight $V(s)$ being prescribed.

  The syntax is more accented towards the first use. For the second use, the user should set $S = 0$, $NS = 0$. This results in (automatically) $SCLR = NSCLR = 0$. The user will also select the `type="input spec"`.

  Let $C_r(s)$ be the reduced order approximation of $C(s)$ which is being sought. Its order is either specified in advance, or the user responds to a prompt after presentation of the weighted Hankel singular values. Then the different types concentrate on (approximately) minimizing certain error measures, through frequency weighted balanced truncation. These are shown in Table 4-1.

Table 4-1. Types versus Error Measures

Type	Error Measure
`"input stab"`	$\left\|[C - C_r]P[I + CP]^{-1}\right\|_\infty$
`"output stab"`	$\left\|[I + PC]^{-1}P[C - C_r]\right\|_\infty$
`"match"`	$\left\|[I + PC]^{-1}P[C - C_r][I + PC]^{-1}\right\|_\infty$
`"match spec"`	$\left\|[I + PC]^{-1}P[C - C_r][I + PC]^{-1}V\right\|_\infty$
`"input spec"`	$\left\|[C - C_r]V\right\|_\infty$

These error measures have certain interpretations, as shown in Table 4-2.

In case $C(s)$ is not a compensator in a closed-loop and the error measure

$$\left\|V(j\omega)[C(j\omega) - C_r(j\omega)]\right\|_\infty$$

is of interest, you can work with `type="input spec"` and C', V' in lieu of C and V.

There is no restriction on the stability of $C(s)$ [or indeed of $P(s)$] in the algorithm, though if $C(s)$ is a controller the closed-loop must be stabilizing. Also, $V(s)$ must be stable. Hence all weights (on the left or right of $C(j\omega) - C_r(j\omega)$ in the error measures) will be stable. The algorithm, however, treats unstable $C(s)$ in a special way, by reducing only the stable part of $C(s)$ (under additive decomposition) and copying the unstable part into $C_r(s)$.

This rather crude approach to the handling of the unstable part of a controller is avoided in `fracred( )`, which provides an alternative to `wtbalance( )` for controller reduction, at least for an important family of controllers.

Table 4-2. Error Measure Interpretation for wtbalance

Type	Error Measure Interpretations
`"input stab"`	A stability robustness argument, based on breaking the loop at the controller output, indicates that if C is stabilizing for P and the error measure is less than 1, then Cr is stabilizing for P. The smaller the error measure is, the greater the stability robustness.
`"output stab"`	A similar stability robustness argument, but based on breaking the loop at the controller input, indicates that if C is stabilizing for P and the error measure is less that 1, then C_r is stabilizing for P. The smaller the error measure is, the greater the stability robustness.
`"match"`	If $T = PC(I + PC)^{-1}$ and $T_r = PC_r(I + PC_r)^{-1}$ are the two closed-loop transfer function matrices, then $T - T_r$ to first order in $C - C_r$ is given by $(I + PC)^{-1}P[C_r - C][I + PC]^{-1}$, so that the error measure looks at matching of the closed-loop transfer function matrix.
`"match spec"`	It may be important to match closed-loop transfer function matrices more at certain frequencies than others; frequency weighting is achieved by introducing $V(s)$. Frequencies corresponding to larger values of $\|V(j\omega)\|$ or $V(j\omega)V_*(j\omega)$ will be the frequencies at which $T(j\omega)$ and $T_r(j\omega)$ should have smaller error.
`"input spec"`	This is the one error measure that is not associated with a plant, or closed-loop of some kind. It simply allows the user to emphasize certain frequencies in the reduction procedures.

Algorithm

The major steps of the algorithm are as follows:

1. Check dimension, syntax, stability of `SysV`, closed-loop stability, and decomposition of $C(s)$ into the sum of a stable part (poles in $Re[s] < 0$) and unstable part (poles in $Re[s] \geq 0$); `stable( )` is used for this purpose.
2. Compute input (right) weight and/or output (left) weight as appropriate for the specified type.

3. Compute weighted Hankel Singular Values σ_i (described in more detail later). If the order of $C_r(s)$ is not specified *a priori*, it must be input at this time. Certain values may be flagged as unacceptable for various reasons. In particular *nscr* cannot be chosen so that $\sigma_{nscr} = \sigma_{nscr+1}$.
4. Execute reduction step on stable part of $C(s)$, based on a modification of `redschur( )` to accommodate frequency weighting, and yielding stable part of $C_r(s)$.
5. Compute $C_r(s)$ by adding unstable part of $C(s)$ to stable part of $C_r(s)$.
6. Check closed-loop stability with $C_r(s)$ introduced in place of $C(s)$, at least in case $C(s)$ is a compensator.

More details of steps 3 and 4, will be given for the case when there is an input weight only. The case when there is an output weight only is almost the same, and the case when both weights are present is very similar, refer to [Enn84a] for a treatment. Let

$$C(s) = D_c + C_c(sI - A_c)^{-1}B_c$$
$$W^S(s) = D_w + C_w(sI - A_w)^{-1}B_w$$

be a stable transfer function matrix to be reduced and its stable weight. Thus, $W(s)$ may be $P(I + CP)^{-1}$, corresponding to `"input stab"`, and will thus have been calculated in step 2; or it maybe an independently specified stable $V(s)$. Then

$$C_s(s)W(s) = D_cD_w + \begin{bmatrix} C_c & D_cC_w \end{bmatrix} \begin{bmatrix} sI - A_c & B_cC_w \\ 0 & sI - A_w \end{bmatrix}^{-1} \begin{bmatrix} B_cD_w \\ B_w \end{bmatrix}$$

The controllability grammian P satisfying

$$P \begin{bmatrix} A_c' & 0 \\ C_w'B_C' & A_w' \end{bmatrix} + \begin{bmatrix} A_c & B_cC_w \\ 0 & A_w \end{bmatrix} P + \begin{bmatrix} B_cD_w \\ B_w \end{bmatrix} \begin{bmatrix} D_w'B_c' & B_w' \end{bmatrix} = 0$$

is written as

$$P = \begin{bmatrix} P_{cc} & P_{cw} \\ P_{cw}' & P_{ww} \end{bmatrix}$$

and the observability grammian Q, defined in the obvious way, is written as

$$Q = \begin{bmatrix} Q_{cc} & Q_{cw} \\ Q_{cw}' & Q_{ww} \end{bmatrix}$$

It is trivial to verify that $Q_{cc}A_c + A_c'Q_{cc} = -C_c'C_c$ so that Q_{cc} is the observability gramian of $C_s(s)$ alone, as well as a submatrix of Q.

The weighted Hankel singular values of $C_s(s)$ are the square roots of the eigenvalues of $P_{cc}Q_{cc}$. They differ from the usual or unweighted Hankel singular values because P_{cc} is not the controllability gramian of $C_s(s)$ but rather a weighted controllability gramian. The usual controllability gramian can be regarded as $E[x_c x_c']$ when $C_s(s)$ is excited by white noise. The weighted controllability gramian is still $E[x_c x_c']$, but now $C_s(s)$ is excited by colored noise, that is, the output of the shaping filter $W(s)$, which is excited by white noise.

Small weighted Hankel singular values are a pointer to the possibility of eliminating states from $C_s(s)$ without incurring a large error in $\left\| [C(j\omega) - C_r(j\omega)]W(j\omega) \right\|_\infty$. No error bound formula is known, however.

The actual reduction procedure is virtually the same as that of `redschur( )`, except that P_{cc} is used. Thus Schur decompositions of $P_{cc}Q_{cc}$ are formed with the eigenvalues in ascending and descending order

$$V_A'P_{cc}Q_{cc}V_A = S_{asc}$$
$$V_D'P_{cc}Q_{cc}V_D = S_{des}$$

The maximum order permitted is the number of nonzero eigenvalues of $P_{cc}Q_{cc}$ that are larger than ε.

The matrices V_A, V_D are orthogonal and S_{asc} and S_{des} are upper triangular. Next, submatrices are obtained as follows:

$$V_{lbig} = V_A \begin{bmatrix} 0 \\ I_{nscr} \end{bmatrix} \qquad V_{rbig} = V_D \begin{bmatrix} I_{nscr} \\ 0 \end{bmatrix}$$

and then a singular value decomposition is formed:

$$U_{ebig}S_{ebig}V_{ebig} = V_{lbig}'V_{rbig}$$

From these quantities the transformation matrices used for calculating $C_{sr}(s)$, the stable part of $C_r(s)$, are defined

$$S_{lbig} = V_{lbig} V_{ebig} S_{ebig}^{-1/2}$$

$$S_{rbig} = V_{rbig} V_{ebig} S_{ebig}^{-1/2}$$

and then

$$AC_R = S'_{lbig} A_C S_{rbig} \quad B_{CR} = S'_{lbig} B_C$$
$$AC_R = C_C S_{rbig} \quad B_{CR} = D_C$$

Just as in unweighted balanced truncation, the reduced order transfer function matrix is guaranteed stable, the same is guaranteed to be true in weighted balanced truncation when either a left (output) weight or a right (input) weight is used. It is suspected to be true when both input and output weights are present. The overall algorithm is not, however, at risk in this case, since it is stability of the closed-loop system which is the key issue of concern, (except for `type="input spec"`, but here there is only a single weight, and so the theory guarantees preservation of stability).

Related Functions

`balance()`, `redschur()`, `stable()`, `fracred()`

fracred( )

```
[SysCR,HSV] = fracred(Sys,Kr,Ke,type,{nscr,Qyy})
```

The `fracred( )` function uses fractional representations to calculate a reduction of a continuous-time compensator comprising a state estimator with state feedback law.

Restrictions

1. The closed-loop system (`SCLR,NSCLR`) is calculated from

   ```
   sysol=scr*sys              # open loop system
   syscl=feedback(sysol)      # closed loop system
   ```

2. Initial state values, state names, and input and output names are not considered by `fracred( )`.

3. Only continuous systems are accepted; for discrete systems use `makecontinuous( )` before calling `bst( )`, then discretize the result.

```
Sys=fracred(makecontinuous(SysD));
SysD=discretize(Sys);
```

Defining and Reducing a Controller

Suppose $P(s) = C(sI - A)^{-1}B$ and $A - BK_R$ and $A - K_EC$ are stable (where K_R is a stabilizing state feedback gain and K_E a stabilizing observer gain). A controller for the plant $P(s)$ can be defined by

$$\begin{aligned}\hat{x} &= A\hat{x} + Bu - K_E(C\hat{x} - y)\\ u &= -K_R\hat{x}\end{aligned}$$

(with u the plant input and y the plant output). The associated series compensator under unity negative feedback is

$$C(s) = K_R(sI - A + BK_R + K_EC)^{-1}K_E$$

and this may be written as a left or right MFD as follows:

$$C(s) = [I + K_R(sI - A + K_EC)^{-1}B]^{-1}K_R(sI - A + K_EC)^{-1}K_E \qquad (4\text{-}5)$$

$$C(s) = K_R(sI - A + BK_R)^{-1}K_E[I + C(sI - A + BK_R)^{-1}K_E]^{-1} \qquad (4\text{-}6)$$

The reduction procedures `"right perf"` and `"left perf"` have similar rationales. We shall describe `"right perf"`, refer to [AnM89] and [LiA86]. The first rationale involves observing that to reduce $C(s)$, one might as well reduce its numerator and denominator simultaneously, and then form a new fraction $C_r(s)$ of lower order than $C(s)$.

This amounts to reducing

$$E(s) = \begin{bmatrix}K_R\\ C\end{bmatrix}(sI - A + BK_R)^{-1}K_E \qquad (4\text{-}7)$$

to, for example,

$$E_r(s) = \begin{bmatrix} \overline{K}_R \\ \overline{C} \end{bmatrix} (sI - \hat{A}) \overline{K}_E$$

through, for example, balanced truncation, and then defining:

$$C(s) = \overline{K}_R (sI - \hat{A})^{-1} \overline{K}_E [I + \overline{C}(sI - \hat{A})^{-1} \overline{K}_E]^{-1}$$

$$= \overline{K}_R (sI - \hat{A} + \overline{K}_E \overline{C})^{-1} \overline{K}_E$$

For the second rationale, consider Figure 4-5.

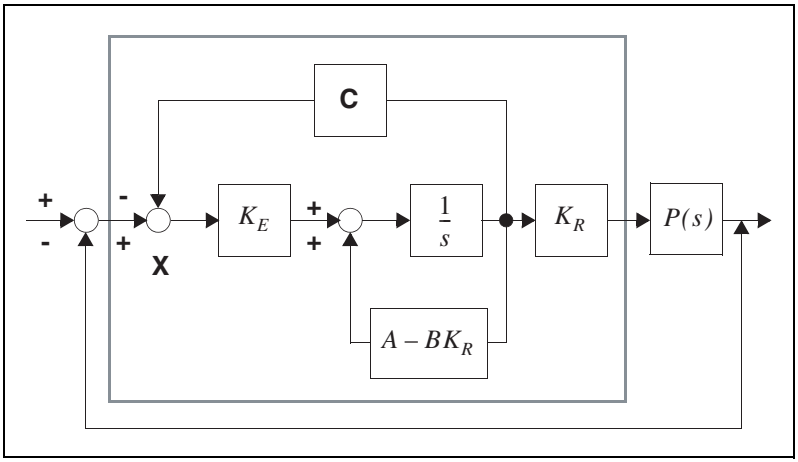

Figure 4-5. Internal Structure of Controller

Recognize that the controller $C(s)$ (shown within the hazy rectangle in Figure 4-5) can be constructed by implementing

$$K_R (sI - A + BK_R)^{-1} K_E$$

and

$$C(sI - A + BK_R)^{-1} K_E$$

and then applying an interconnection rule (connect the output of the second transfer function matrix back to the input at point **X** in Figure 4-5).

Controller reduction proceeds by implementing the same connection rule but on reduced versions of the two transfer function matrices.

When K_E has been defined through Kalman filtering considerations, the spectrum of the signal driving K_E in Figure 4-5 is white, with intensity Q_{yy}. It follows that to reflect in the multiple input case the different intensities on the different scalar inputs, it is advisable to introduce at some stage a weight $Q_{yy}^{1/2}$ into the reduction process.

Algorithm

After preliminary checks, the algorithm steps are:

1. Form the observability and weighted (through Q_{yy}) controllability grammians of $E(s)$ in Equation 4-7 by

$$P(A - BK_R)' + (A - BK_R)P = -K_E Q_{yy} K_E' \tag{4-8}$$

$$Q(A - BK_R) + (A - BK_R)'Q = -K_R' K_R - C'C \tag{4-9}$$

2. Compute the square roots of the eigenvalues of PQ (Hankel singular values of the fractional representation of Equation 4-5). The maximum order permitted is the number of nonzero eigenvalues of PQ that are larger than ε.
3. Introduce the order of the reduced-order controller, possibly by displaying the Hankel singular values (HSVs) to the user. Broadly speaking, one can throw away small HSVs but not large ones.
4. Using `redschur( )`-type calculations, find a state-variable description of $E_r(s)$. This means that $E_r(s)$ is the transfer function matrix of a truncation of a balanced realization of $E(s)$, but the `redschur( )` type calculations avoid the possibly numerically difficult step of balancing the initially known realization of $E(s)$. Suppose that:

$$\hat{A} = S_{lbig}'(A - BK_R)S_{rbig}, \overline{K}_E = S_{lbig}' K_E$$

5. Define the reduced order controller $C_r(s)$ by

$$A_{CR} = S_{lbig}'(A - BK_R - K_E C)S_{rbig} \tag{4-10}$$

so that

$$C_r(s) = C_{CR}(sI - A_{CR})^{-1} B_{CR}$$

6. Check the stability of the closed-loop system with $C_r(s)$. When the `type="left perf"` is specified, one works with

$$E(s) = K_R(sI - A + K_E C)^{-1}\begin{bmatrix} B & K_E \end{bmatrix} \tag{4-11}$$

which is formed from the numerator and denominator of the MFD in Equation 4-5. The grammian equations (Equation 4-8 and Equation 4-9) are replaced by

$$P(A - K_E C)' + (A - K_E C)P = -BB' - K_E K_E'$$

$$Q(A - K_E C) + (A - K_E C)'Q = -K_R' K_R$$

`redschur( )`-type calculations are used to reduce $E(s)$ and Equation 4-10 again yields the reduced-order controller. Notice that the HSVs obtained from Equation 4-10 or the left MFD (Equation 4-5) of $C(s)$ will in general be quite different from those coming from the right MFD (Equation 4-6). It may be possible to reduce much more with the left MFD than with the right MFD (or vice-versa) before closed-loop stability is lost.

As noted in the `fracred( )` input listing, `type="left stab"` and `"right stab"` focus on a stability robustness measure, in conjunction with Equation 4-5 and Equation 4-6, respectively. Leaving aside for the moment the explanation, the key differences in the algorithm computations lie solely in the calculation of the grammians P and Q. For `type="left stab"`, these are given by

$$P(A - BK_R)' + (A - BK_R)P = -BB'$$

$$Q(A - K_E C) + (A - K_E C)'Q = -K_R' K_R$$

and for `"right stab"`,

$$P(A - BK_R)' + (A - BK_R)P = -K_E K_E' \tag{4-12}$$

$$Q(A - K_E C) + (A - K_E C)'Q = -C'C \tag{4-13}$$

Additional Background

A discussion of the stability robustness measure can be found in [AnM89] and [LAL90]. The idea can be understood with reference to the transfer functions $E(s)$ and $E_r(s)$ used in discussing `type="right perf"`. It is possible to argue (through block diagram manipulation) that

- $C(s)$ stabilizes $P(s)$ when $E(s)$ stabilizes (as a series compensator) with unity negative feedback $\hat{P}(s) = \begin{bmatrix} P(s) & I \end{bmatrix}$.
- $E_r(s)$ also will stabilize $[P(s)I]$, and then $C_r(s)$ will stabilize $P(s)$, provided

$$\left\| \begin{bmatrix} C(j\omega I - A + K_E C)^{-1} B & I - C(j\omega I - A + K_E C)^{-1} K_E \\ [E(j\omega) - E_e(j\omega)] \end{bmatrix} \right\|_\infty < 1 \quad (4\text{-}14)$$

Accordingly, it makes sense to try to reduce E by frequency-weighted balanced truncation. When this is done, the controllability grammian for $E(s)$ remains unaltered, while the observability grammian is altered. (Hence Equation 4-5, at least with $Q_{yy} = I$, and Equation 4-12 are the same while Equation 4-6 and Equation 4-13 are quite different.) The calculations leading to Equation 4-13 are set out in [LAL90].

The argument for `type="left perf"` is dual. Another insight into Equation 4-14 is provided by relations set out in [NJB84]. There, it is established (in a somewhat broader context) that

$$\{ C(j\omega I - A + K_E C)^{-1} B \qquad I - C(j\omega I - A + K_E C)^{-1} K_E \}$$
$$\times \begin{bmatrix} K_r(sI - A + BK_R)^{-1} K_E \\ I + C(j\omega I - A + BK_R)^{-1} K_E \end{bmatrix} = I$$

The left matrix is the weighting matrix in Equation 4-14; the right matrix is the numerator of $C(j\omega)$ stacked on the denominator, or alternatively

$$\mathrm{E}(j\omega) + \begin{bmatrix} 0 \\ I \end{bmatrix}$$

This formula then suggests the desirability of retaining the weight in the approximation of $E(j\omega)$ by $E_r(j\omega)$.

The four schemes all produce different HSVs; it follows that it may be prudent to try all four schemes for a particular controller reduction. Recall again that their relative sizes are only a guide as to what can be thrown away without incurring much error. There is no simple rule to indicate which of the four schemes will be the most effective at controller reduction.

Two rough rules can, however, be formulated.

- Problems with instability through reduction to too low a controller order are more likely with `"left perf"` and `"right perf"` than `"left stab"` or `"right stab"`.
- If the controller has been designed using the loop transfer recovery idea, `"left stab"` will probably be attractive if the input noise covariance is very large, and `"right stab"` will probably be attractive if the output weighting in the performance index is very large, [LiA90]. The reduced controllers will then actually be very similar to those obtained using `wtbalance( )` with the option `"input stab"` in the first case and `"output stab"` in the second case.

One example gives the HSVs summarized in Table 4-3 for an eighth order controller.

Table 4-3. HSVs for an Eighth Order Controller

	1	2	3	4	5	6	7	8
`right perf`	.0339	.0164	.0128	.0102	.0040	.0037	.0000	.0000
`left perf`	4.9075	4.8742	3.8457	3.7813	1.2255	1.1750	.5055	.0413
`right stab`	3.3081	.7278	.1123	.0783	.0242	.0181	.0107	.0099
`left stab`	1.3914	1.317	1.1269	1.0862	.9638	.5846	.5646	.3144

The most attractive candidate for reducing to second order is `right stab`. This is because the HSVs being discarded (columns 3 to 8) are smaller relative to those being retained (columns 1 and 2) for `right stab` than for the other three candidates.

Note The relative values count, not the absolute values.

Related Functions

`redschur()`, `wtbalance()`

5

Utilities

This chapter describes three utility functions: `hankelsv( )`, `stable( )`, and `compare( )`.

The background to `hankelsv( )`, which calculates Hankel singular values, was presented in Chapter 1, *Introduction*. Hankel singular values are also calculated in other functions, sometimes by other procedures. A comparison of the procedures is given in the *Hankel Singular Values* section. The function `compare( )` serves to facilitate the comparisons of an unreduced and a reduced system, from various points of views.

The function `stable( )` is used to separate (additively) a system into its stable and unstable parts, that is, given $G(s)$, the function determines $G_s(s)$ and $G_u(s)$, the first with all poles in $Re[s] < 0$, the second with all poles in $Re[s] \geq 0$, such that

$$G(s) = G_s(s) + G_u(s)$$

The function is used within some of the other functions of the Model Reduction Module. It should also be used when reduction of an unstable $G(s)$ is contemplated. The normal reduction functions, for example, `balmoore( )` or `redschur( )`, require stability of the transfer function matrix $G(s)$ being reduced. If $G(s)$ is unstable, `stable( )` should be used to generate $G_s(s)$ and $G_u(s)$; reduction of $G_s(s)$ should be performed, and then $G_u(s)$ added to the outcome using the + operator, to yield the desired reduction of $G_s(s)$.

hankelsv( )

```
[HSV,Wc,Wo] = hankelsv(Sys,{noplot})
```

The `hankelsv( )` function computes the Hankel Singular Values of a stable system (continuous or discrete) and displays them in a bar plot.

The gramian matrices are defined by solving the equations (in continuous time)

$$AW_c + W_cA' = -BB'$$
$$W_oA + A'W_0 = -C'C$$

and, in discrete time

$$W_c - AW_cA' = BB'$$
$$W_o - A'W_oA = C'C$$

The computations are effected with `lyapunov( )` and stability is checked, which is time-consuming. The Hankel singular values are the square roots of the eigenvalues of the product.

Related Functions

`lyapunov()`, `dlyapunov()`

stable( )

```
[SysS,SysU] = stable(Sys,{tol})
```

The `stable( )` function decomposes `Sys` into its stable (`SysS`) and unstable(`SysU`) parts, such that `Sys=SysS+SysU`.

Continuous systems have unstable poles if real parts > `-tol`.

Discrete systems have unstable poles if magnitudes > `1-tol`.

- The direct term (D matrix) is included in `SysS`.
- If `Sys` has poles clustered near `-tol` (or `1-tol`), then `SysS` and `SysU` might be ill-conditioned. To avoid this problem choose `tol` to a value that is not close to the majority of poles.

Algorithm

The algorithm begins by transforming the A matrix to Schur form, and counting the number of stable and unstable eigenvalues, together with those for which classification is doubtful. Stable eigenvalues are those in either of the following:

- $Re[s] < 0$ (continuous time)
- $|z| < 1$ (discrete time)

Doubtful ones are those for which the real part of the eigenvalue has magnitude less than or equal to `tol` for continuous-time, or eigenvalue magnitude within the following range for discrete time:

$$1 - tol,\ 1 + tol$$

A warning is given if doubtful eigenvalues exist.

The algorithm then computes a real ordered Schur decomposition of A so that after transformation

$$A = \begin{bmatrix} A_S & A_{SU} \\ 0 & A_U \end{bmatrix}$$

where the eigenvalues of A_S and A_U are respectively stable and unstable. A matrix X satisfying $-A_{SX} + XA_U + A_{SU} = 0$ is then determined by calling the algorithm `sylvester( )`. The eigenvalue properties of A_S and A_U guarantee that X exists. If doubtful eigenvalues are present, they are assigned to the unstable part of `Sys`. In this circumstance you get the message,

`The system has poles near, or upon, the jw-axis`

for continuous systems, and the following for discrete systems:

`The system has poles near the unit circle.`

Note If A has eigenvalues clustered near `-tol` (`1-tol` in discrete-time), then X is likely to be ill-conditioned and consequently `SysS` and `SysU` will also be ill-conditioned. (For example, the B matrix of `SysS` could contain very small values, while the C matrix could contain large values. In this case, `SysS` would be very weakly controllable and very strongly observable. This will cause problems when gramians and Hankel singular values are calculated.) To avoid this problem, change `tol` to a value that is not close to the majority of eigenvalues.

A further transformation of A is constructed using X:

$$A \rightarrow \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} \qquad A \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix}$$

$$= \begin{bmatrix} A_S & 0 \\ 0 & A_U \end{bmatrix}$$

After this last transformation, and with

$$B = \begin{bmatrix} B_S \\ B_U \end{bmatrix} \qquad C = [C_S C_U]$$

it follows that

$$SysS = [A_S, A_S; C_S D]$$

and

$$SysU = [A_U, B_U; C_U 0]$$

By combining the transformation yielding the real ordered Schur form for A with the transformation defined using X, the overall transformation T is readily identified. In case all eigenvalues of A are stable or all are unstable, this is flagged, and T = I.

`stable( )` can be combined with a reduction algorithm such as `redschur( )` or `balmoore( )` to reduce the order of a system with some unstable and some stable poles. One uses `stable( )` to separate the stable and unstable parts, and then, for example, reduces the stable part with `redschur( )`; the reduced stable part is added to the original unstable part to provide the desired system reduction.

Related Functions

`sylvester()`, `schur()`, `redschur()`, `balmoore()`

compare( )

```
[respdiff] = compare(Sys,SysRed,FTvec,{Fmin,Fmax,npts,radians,type})
```

The `compare( )` function provides a number of different graphical tests which can be used to compare two state-space system implementations. `compare( )` can be used as a tool for evaluating a reduced-order system by comparing it with the original full-order system from which it was obtained. However, it can be used for more general comparisons as well, such as examining the results of different discretization or identification techniques.

6

Tutorial

This chapter illustrates a number of the MRM functions and their underlying ideas. A plant and full-order controller are defined, and then the effects of various reduction algorithms are examined. The data for this example is stored in the file `mr_disc.xmd` in the Xmath demos directory. To follow the example, start Xmath, and then select **File»Load** from the **Xmath Commands** menu, or enter the load command with the file specification appropriate to your operating system from the **Xmath Commands** area. For example:

```
load "$XMATH/demos/mr_disc"
```

Plant and Full-Order Controller

The plant in question comprises four spinning disks, connected by a flexible shaft. A motor applies torque to the third disk, and the output variable of interest is the angular displacement of the first disk. The plant transfer function, which is nonminimum phase and has a double pole at the origin, is as follows:

$$G(s) = \frac{1}{4s^2} \frac{\frac{s^2 2\zeta_0\omega_0 s + \omega_0^2}{\omega_0^2} \cdot \frac{s^2\zeta_1\omega_1 s + \omega_1^2}{\omega_1^2} \cdot \frac{s+a}{a}}{\frac{s^2 2\zeta_2\omega_2 s + \omega_2^2}{\omega_2^2} \cdot \frac{s^2 2\zeta_3\omega_3 s + \omega_3^2}{\omega_3^2} \cdot \frac{s^2 2\zeta_4\omega_4 s + \omega_4^2}{\omega_4^2}}$$

with:

ζ_0=0.02 ω_0=1
ζ_1=-0.4 ω_1=5.65
ζ_2=0.02 ω_2=0.765
ζ_3=0.02 ω_3=1.41
ζ_4=0.02 ω_4=1.85
a=4.84

A minimal realization in modal coordinates is $C(sI - A)^{-1}B$ where:

$$A = diag\left\{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} -0.015 & 0.765 \\ -0.765 & -0.015 \end{bmatrix}, \begin{bmatrix} -0.028 & 1.410 \\ -1.410 & -0.028 \end{bmatrix}, \begin{bmatrix} -0.04 & 1.85 \\ -1.85 & -0.04 \end{bmatrix}\right\}$$

$$B = \begin{bmatrix} 0.026 \\ -0.251 \\ 0.033 \\ -0.886 \\ -4.017 \\ 0.145 \\ 3.604 \\ 0.280 \end{bmatrix} \qquad C' = \begin{bmatrix} -0.996 \\ -0.105 \\ 0.261 \\ 0.009 \\ -0.001 \\ -0.043 \\ 0.002 \\ -0.026 \end{bmatrix}$$

The specifications seek high loop gain at low frequencies (for performance) and low loop gain at high frequencies (to guarantee stability in the presence of unstructured uncertainty). More specifically, the loop gain has to lie outside the shaded region shown in Figure 6-1.

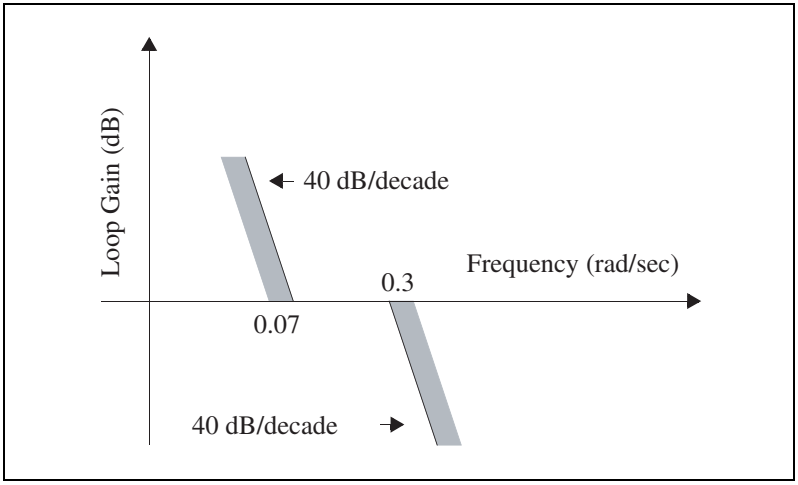

Figure 6-1. Loop Gain Constraints

With a state weighting matrix,

```
Q = 1e-3*diag([2,2,80,80,8,8,3,3]);
R = 1;
```

(and unity control weighting), a state-feedback control-gain is determined through a linear-quadratic performance index minimization as:

```
[Kr,ev] = regulator(sys,Q,R);
```

$A - B \times K_r$ is stable. Next, with an input noise variance matrix $Q = W_t BBW_t$ where,

$$W_t = DIAG([0.346, 0.346, 0.024, 0.0240.042, 0.0420.042, 0.042])$$

and measurement noise covariance matrix $\hat{R}$=1, an estimation gain K_e (so that $A - K_e C$ is stable) is determined:

```
Qhat = Wt*b*b'*Wt;
Rhat = 1;
[Ke,ev] = estimator(sys,Qhat,Rhat,{skipChks});
```

The keyword `skipChks` circumvents syntax checking in most functions. It is used here because we know that `Qhat` does not fulfill positive semidefiniteness due to numerics).

```
sysc=lqgcomp(sys,Kr,Ke);
poles(sysc)
ans (a column vector) =

  -0.296674 + 0.292246 j
  -0.296674 - 0.292246 j
  -0.15095  + 0.765357 j
  -0.15095  - 0.765357 j
  -0.239151 + 1.415    j
  -0.239151 - 1.415    j
  -0.129808 + 1.84093  j
  -0.129808 - 1.84093  j
```

The compensator itself is open-loop stable. A brief explanation of how `Q` and `Wt` are chosen is as follows. First, `Q` is chosen to ensure that the loop gain $\left|K_r(j\omega I - A)^{-1}B\right|$ (which would be relevant were the state measurable) meets the constraints as far as possible. However, it is not possible to obtain a 40 dB per decade roll-off at high frequencies, as *LQ* design virtually always yields a 20 dB per decade roll-off. Second, a loop transfer recovery approach to the choice of $\hat{Q}$ as $\rho BB'$ for some large ρ is modified through the introduction of the diagonal matrix `Wt`. The larger entries of `Wt`, because of the modal coordinate system, in effect promote better loop transfer

recovery at low frequencies; there is consequently a faster roll-off of the loop gain at high frequencies than for $\left|K_r(j\omega I - A)^{-1}B\right|$, and this is desired.

Figure 6-2 displays the (magnitudes of the) plant transfer function, the compensator transfer function and the loop gain, as well as the constraints; evidently the compensated plant meets the constraints.

You can enter the following commands to create a plot equivalent to Figure 6-2:

```
sysol=sys*sysc;
svals=svplot(sys,w,{radians});
svalsc=svplot(sysc,w,{radians});
svalsol=svplot(sysol,w,{radians});
plot(svals,{x_log,!grid,!ylab,
line_width=2,hold})
plot(svalsc,{keep})
plot(svalsol,{keep})
f2=plot(wc,constr,{keep,
legend=["plant","compensator",
"compensated plant","constraint"]})
plot({!hold})
```

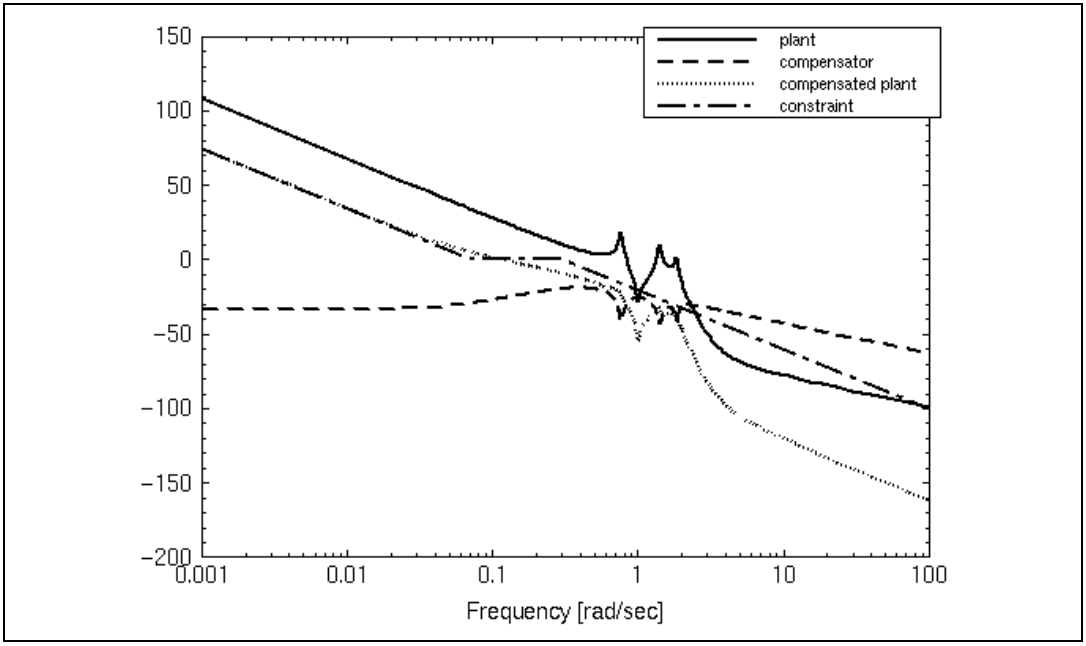

Figure 6-2. Frequency Response for Plant, Compensator, and Compensated Plant

Controller Reduction

This section contrasts the effect of unweighted and weighted controller reduction. Unweighted reduction is at first examined, through `redschur( )` (using `balance( )` or `balmoore( )` will give similar results). The Hankel singular values of the controller transfer function are

6.264×10^{-2} 4.901×10^{-2} 2.581×10^{-2} 2.474×10^{-2}

1.545×10^{-2} 1.335×10^{-2} 9.467×10^{-3} 9.466×10^{-3}

A reduction to order 2 is attempted. The ending Hankel singular values, that is, σ_3, σ_4, ..., σ_8, have a sum that is *not* particularly small with respect to σ_1 and σ_2; this is an indication that problems may arise in the reduction.

```
[syscr,hsv] = redschur(sysc,2);
svalsRol = svplot(sys*syscr,w,{radians});
plot(svalsol, {keep})
f3=plot(wc, constr,{keep,!grid,
legend=["reduced","original","constrained"],
title="Open-Loop Gain Using redschur()"})
```

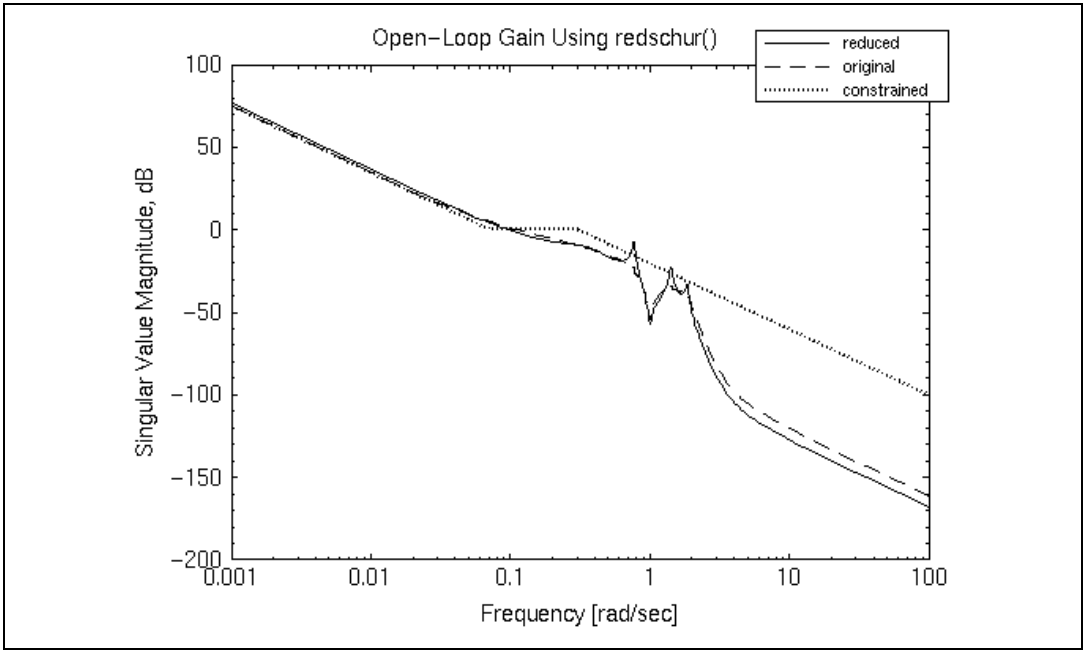

Figure 6-3. Open-Loop Gain Using redschur

Figures 6-3, 6-4, and 6-5 display the outcome of the reduction. The loop gain is shown in Figure 6-3. The error near the unity gain crossover frequency may not look large, but it is considerably larger than that obtained through frequency weighted reduction methods, as described later.

Figure 6-3 also shows the inability to suppress all three plant resonances, in contrast to the full-order controller. Two are such as to cause violation of the specifications. The closed-loop gains differ by some 4 to 5 dB between the full-order and reduced-order controller, in the vicinity of 0.1 radians per second. The step response has overshoot of 50% as opposed to 40% and the ripple persists for longer.

We use the `compare( )` function (refer to the *compare( )* section of Chapter 5, *Utilities*) to reproduce Figures 6-4 and 6-5. Calculate the full-order closed-loop system, then the closed-loop system with the reduced-order compensator:

```
syscl = feedback(sysol);
sysolr=sys*syscr;
sysclr=feedback(sysolr);
```

Generate Figure 6-4:

```
compare(syscl,sysclr,w,{radians,type=5})
f4=plot({keep,legend=["original","reduced"]})
```

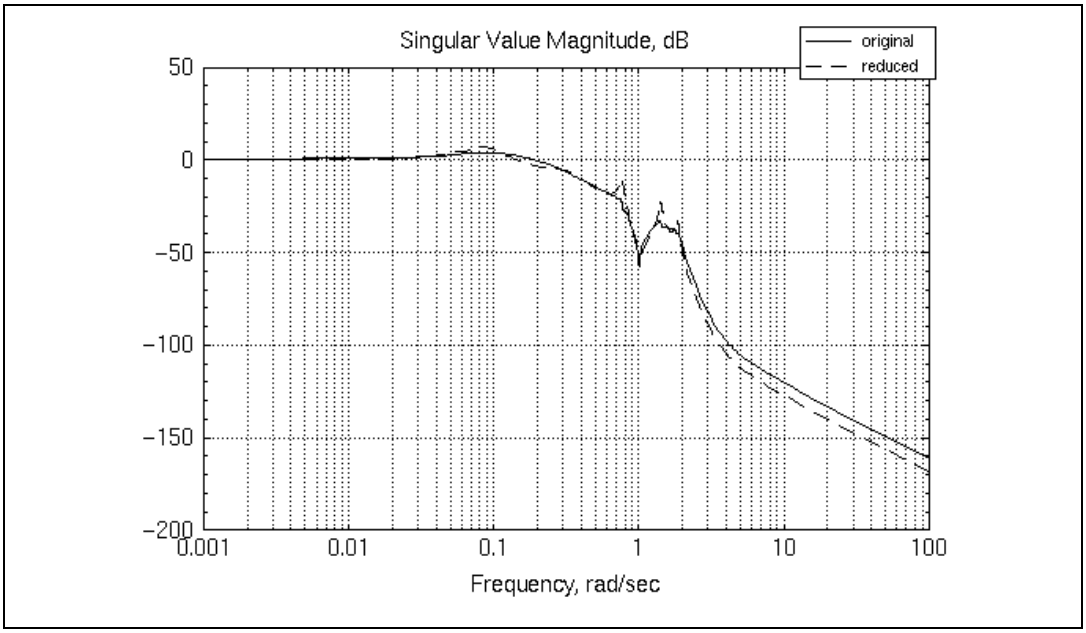

Figure 6-4. Closed-Loop Gain with redschur

Generate Figure 6-5:

```
tvec=0:(140/99):140;
compare(syscl,sysclr,tvec,{type=7})
f5=plot({keep,legend=["original","reduced"]})
```

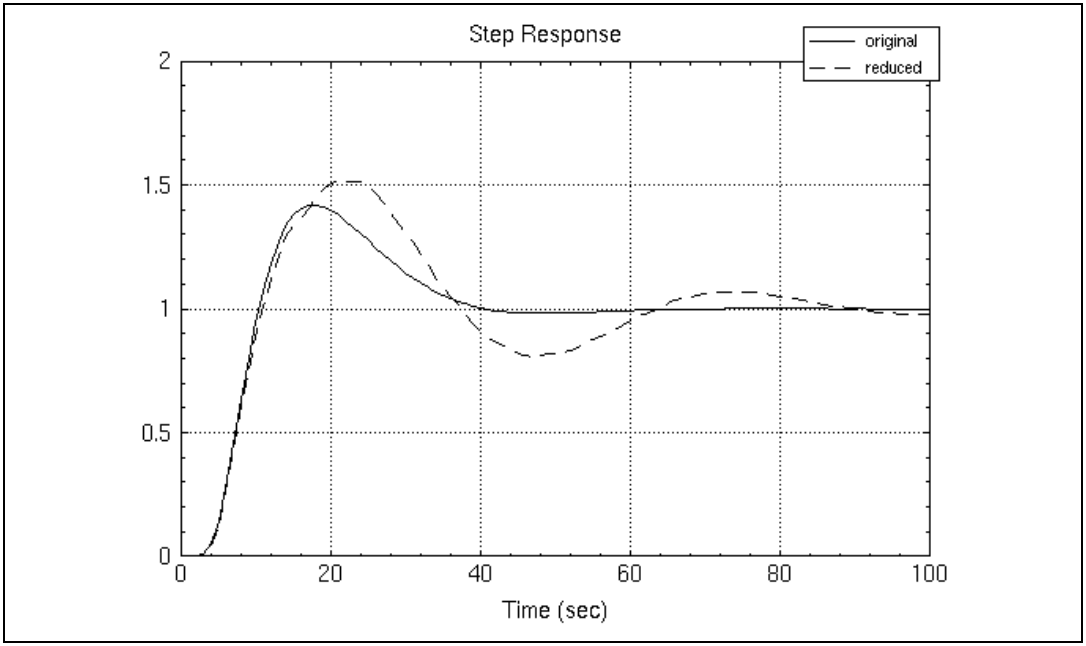

Figure 6-5. Step Response with redschur

ophank( )

`ophank( )` is next used to reduce the controller with the results shown in Figures 6-6, 6-7, and 6-8.

Generate Figure 6-6:

```
[syscr,sysu,hsv]=ophank(sysc,2);
svalsrol = svplot(sys*syscr,w,{radians});
plot(svalsol, {keep})
f6=plot(wc, constr, {keep,!grid,
title="Open-loop gain using ophank()"})
```

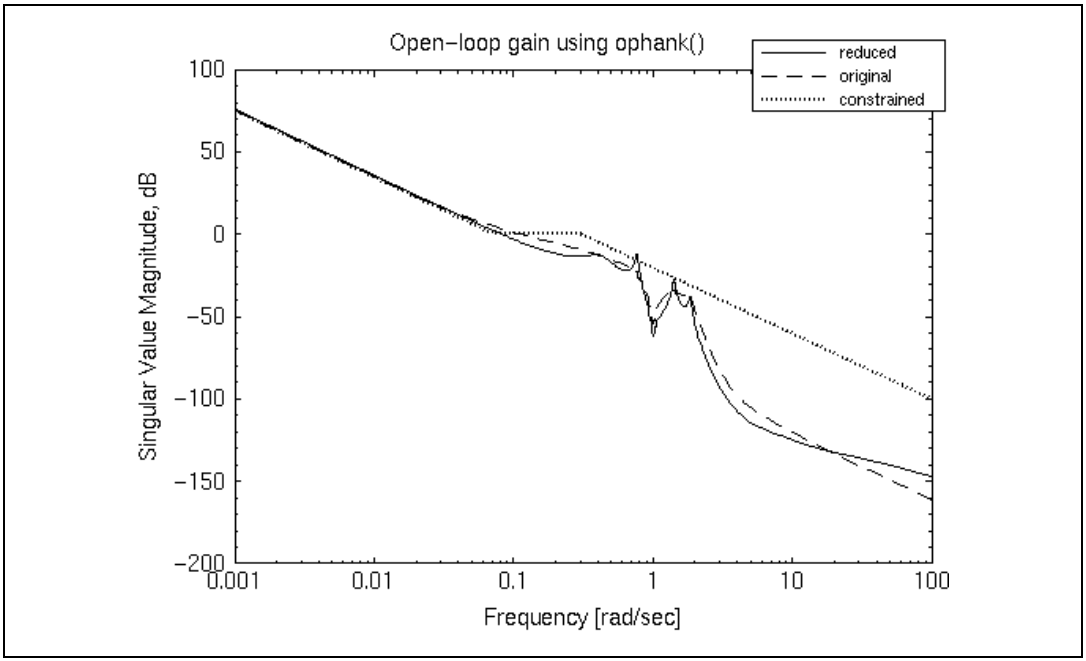

Figure 6-6. Open-Loop Gain Using ophank

Generate Figure 6-7:

```
syscl = feedback(sysol);
sysolr=sys*syscr;
sysclr=feedback(sysolr);
compare(syscl,sysclr,w,{radians,type=5})
f7=plot({keep,legend=["original","reduced"]})
```

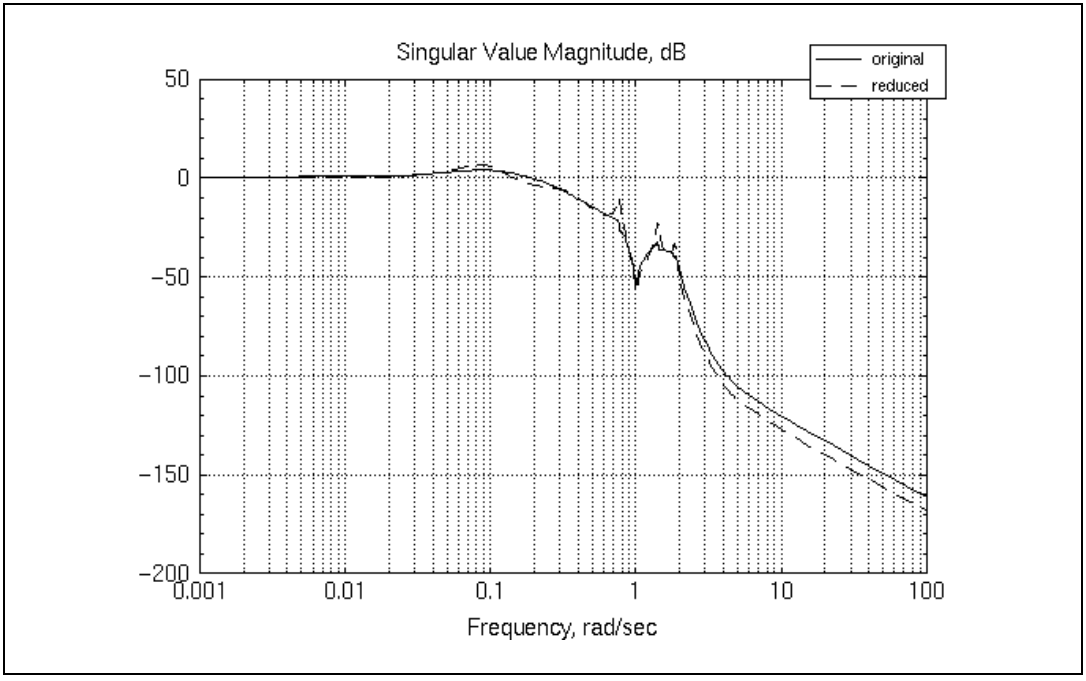

Figure 6-7. Closed-Loop Gain with ophank

Generate Figure 6-8:

```
tvec=0:(140/99):140;
compare(syscl,sysclr,tvec,{type=7})
f8=plot({keep,legend=["original","reduced"]})
```

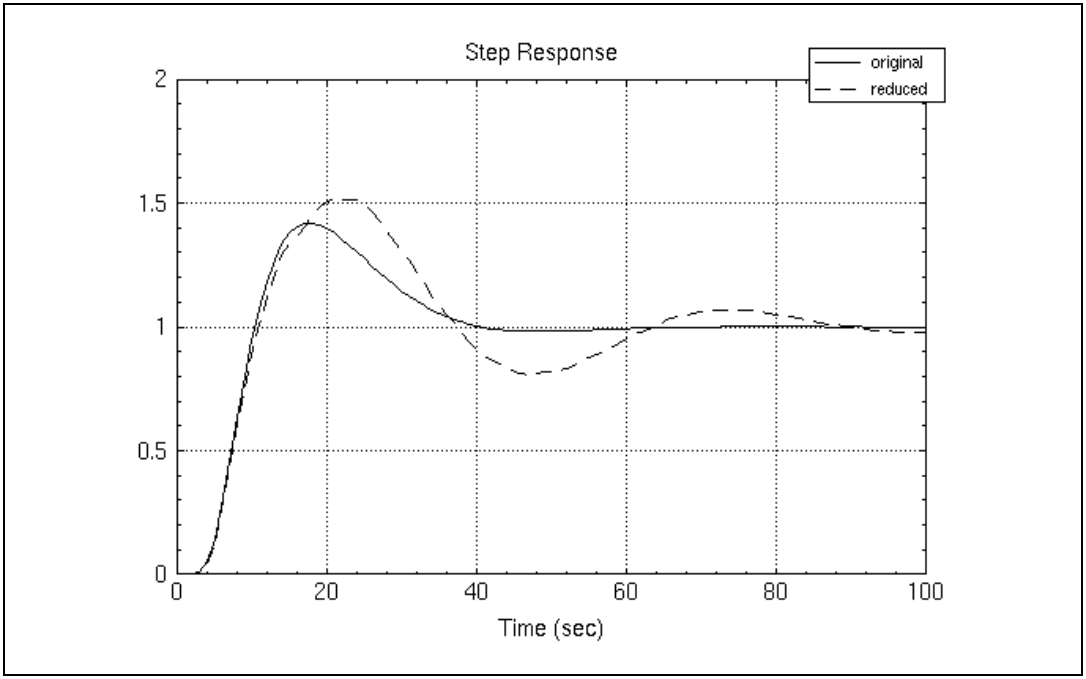

Figure 6-8. Step Response with ophank

The open-loop gain, closed-loop gain and step response are all inferior to those obtained with `redschur( )`. This emphasizes the point that one cannot automatically assume that, because the error bound formula for `ophank( )` is more attractive than that for `redschur( )`, the error itself will be better for `ophank( )`.

wtbalance

The next command examined is `wtbalance` with the option `"match"`.

```
[syscr,ysclr,hsv] = wtbalance(sys,sysc,"match",2)
```

Recall that this command should promote matching of closed-loop transfer functions. The weighted Hankel singular values are:

$$1.486\ 4.513 \times 10^{-1}\ 8.420 \times 10^{-2}\ 5.869 \times 1^{-2}$$

$$1.999 \times 10^{-2}\ 1.382 \times 10^{-2}\ 7.198 \times 10^{-3}\ 6.336 \times 10^{-3}$$

The relative magnitudes suggest that reduction to order 2 will produce less of an approximation error here (in the closed-loop transfer function) than a reduction to this order through `redschur( )` or `ophank( )` (where the implicit criterion is the unweighted error in approximating the controller transfer function). Examination of Figures 6-9, 6-10, and 6-11 reveals that far better approximation is now obtained.

Violation of the specification is to be observed in the open-loop gain. Notice though that:

- The error measure for `wtbalance` does *not* reflect the open-loop gain; it reflects the closed-loop gain.
- While the error in dB looks large, as an absolute value it is not extremely so; `wtbalance` works with additive, not multiplicative error.

Hence, it cannot be concluded that the algorithm is not working. Use of the option `"match spec"` with `wtbalance` might be conjectured as a device for reducing the violation of the specification: one could introduce a weight $V(jw)$ emphasizing frequencies from 0.1 radians per second to 5 radians per second.

For example,

$$V(j\omega) = \frac{(s+0.1)(s+10)}{(s+1)(s+1.4)}$$

This would tend to force the closed-loop transfer functions derived from the full-order and reduced controller to match better over this range; because their absolute value is small there, they are approximately equal to the open-loop gains which, accordingly, may be close. The flaw in this reasoning is that a second-order controller, with four independent parameters only, can only do so much, and the totality of designer demands cannot be fully met.

The following function calls produce Figure 6-9:

```
svalsrol = svplot(sys*syscr,w,{radians})
plot(svalsol, {keep})
f9=plot(wc, constr, {keep,!grid,
legend=["reduced","original","constrained"],
title="Open-Loop Gain Using wtbalance()"})
```

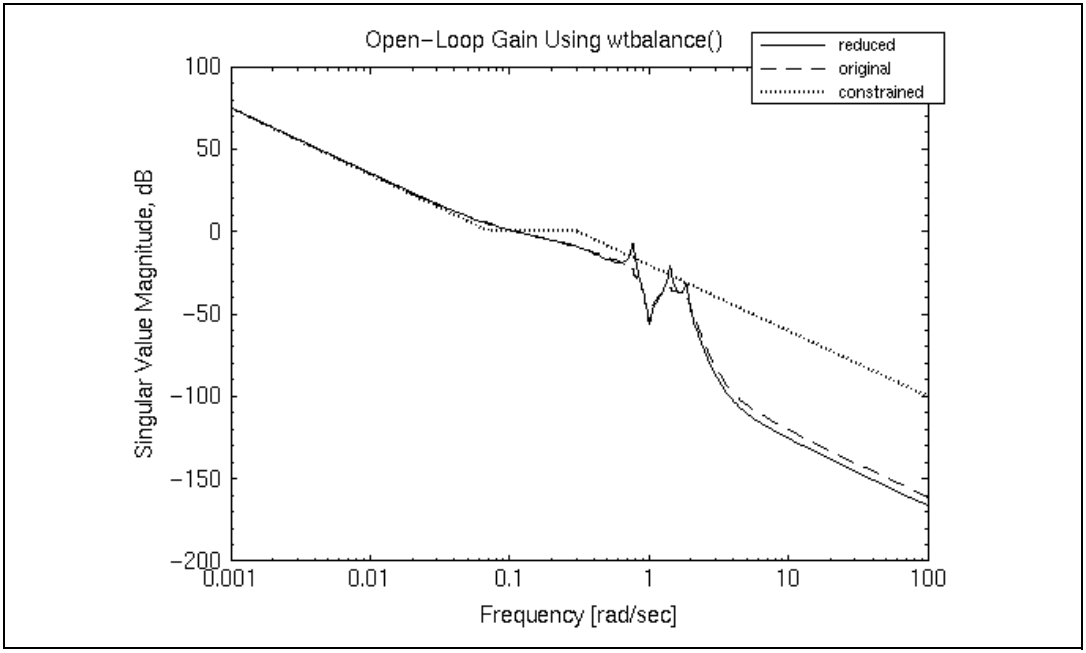

Figure 6-9. Open-Loop Gain with wtbalance

Generate Figure 6-10:

```
syscl = feedback(sysol);
sysolr=sys*syscr;
sysclr=feedback(sysolr);
compare(syscl,sysclr,w,{radians,type=5})
f10=plot({keep,legend=["original","reduced"]})
```

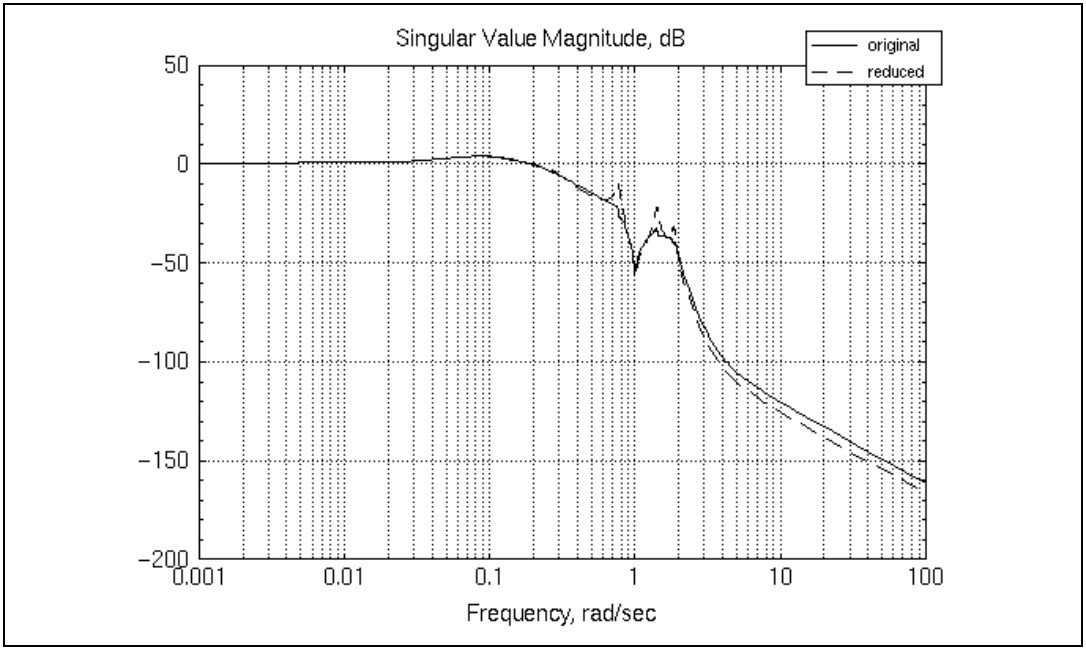

Figure 6-10. Closed-Loop Gain with wtbalance

Generate Figure 6-11:

```
tvec=0:(140/99):140;
compare(syscl,sysclr,tvec,{type=7})
f11=plot({keep,legend=["original","reduced"]})
```

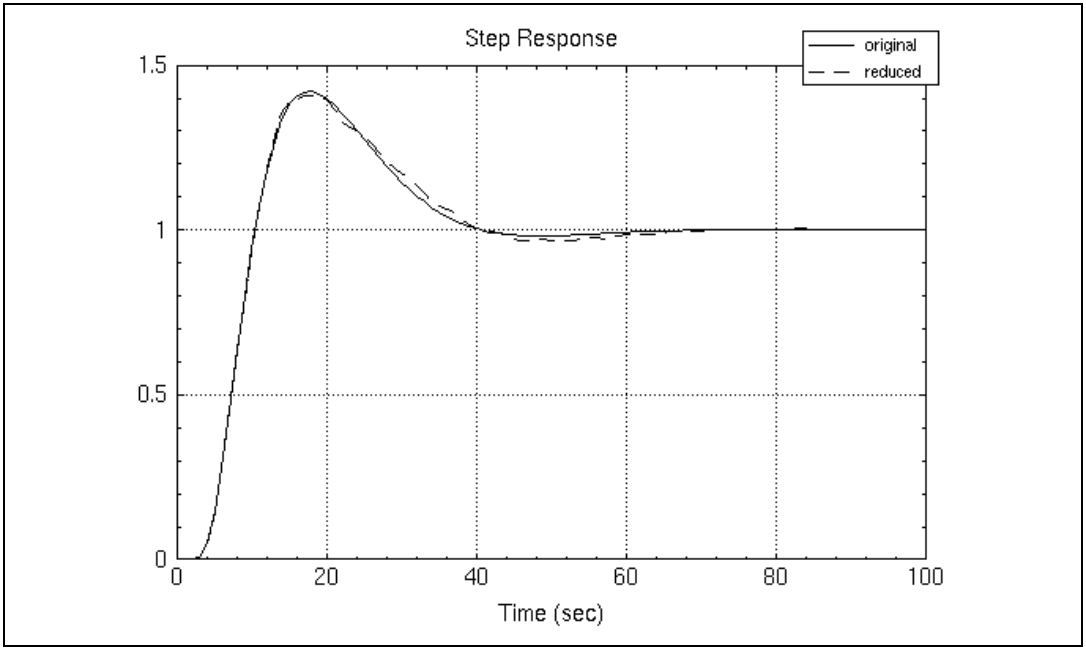

Figure 6-11. Step Response with wtbalance

Figures 6-9, 6-10, and 6-11 are obtained for `wtbalance` with the option `"input spec"`. Evidently, there is little difference between this and the result with the option `"match"`. One notices marginally better matching in the region of interest (0.1 to 5 rad per second) at the expense of matching at other frequencies. The weighted Hankel singular values again indicate that it is reasonable to seek a second order controller.

Generate Figure 6-12:

```
vtf=poly([-0.1,-10])/poly([-1,-1.4])
[,sysv]=check(vtf,{ss,convert});
svalsv = svplot(sysv,w,{radians});
```

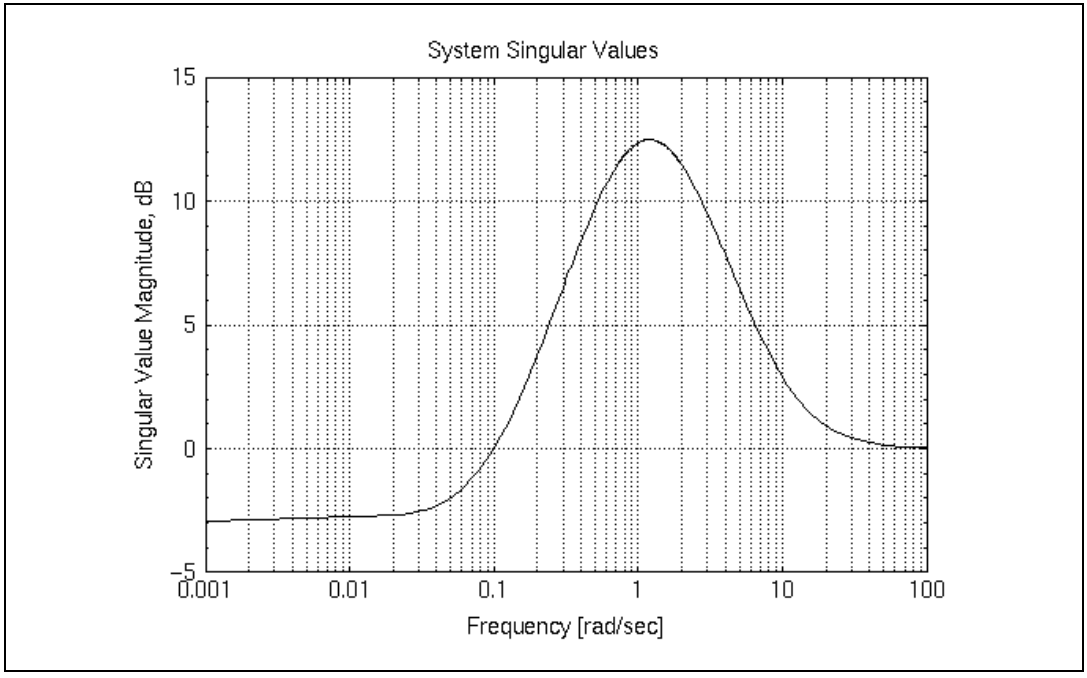

Figure 6-12. Frequency Response of the Weight V($j\omega$)

Generate Figure 6-13:

```
[syscr,sysclr,hsv] = wtbalance(sys,sysc,
"input spec",2,sysv)
svalsrol = svplot(sys*syscr,w,{radians})
plot(svalsol, {keep})
f13=plot(wc,constr,{keep, !grid,
legend=["reduced","original","constrained"],
title="Open-Loop Gain with wtbal(), \"input spec\""})
```

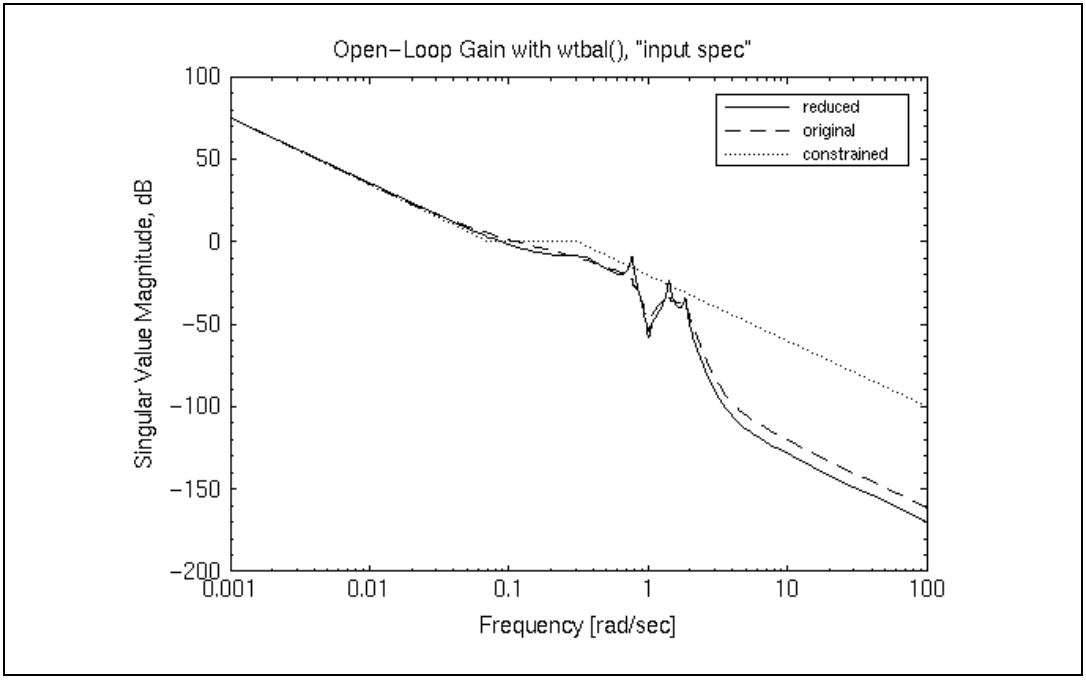

Figure 6-13. Open-Loop Gain from wtbalance with "input spec"

Generate Figure 6-14:

```
syscl = feedback(sysol);
sysolr=sys*syscr;
sysclr=feedback(sysolr);
compare(syscl,sysclr,w,{radians,type=5})
f14=plot({keep,legend=["original","reduced"]})
```

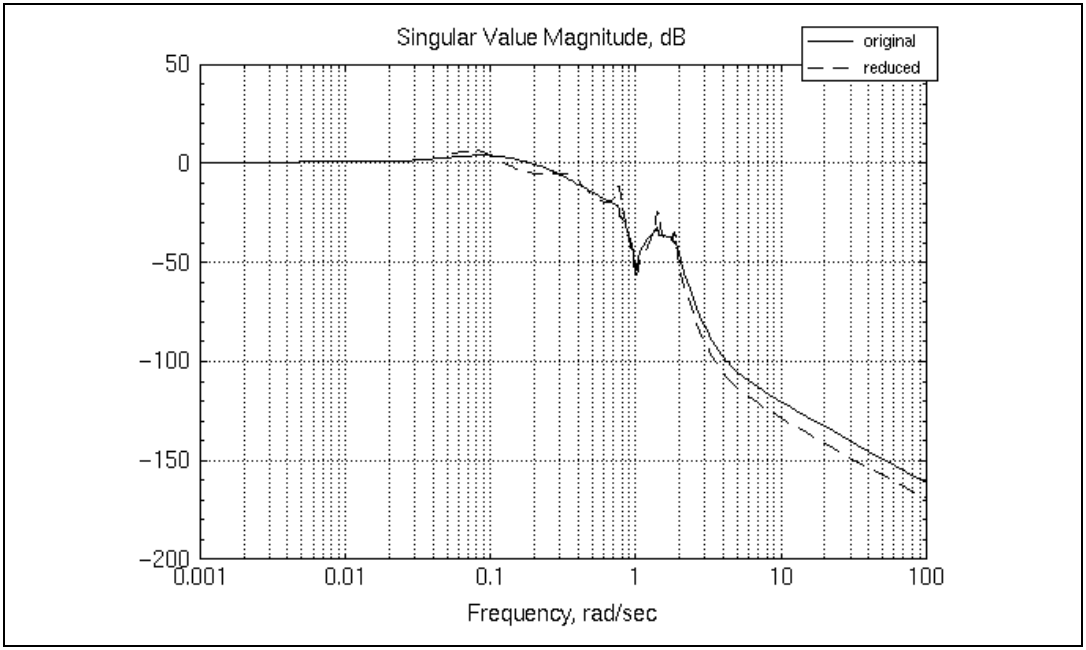

Figure 6-14. System Singular Values of wtbalance with "input spec"

Generate Figure 6-15:

```
tvec=0:(140/99):140;
compare(syscl,sysclr,tvec,{type=7})
f15=plot({keep,legend=["original","reduced"]})
```

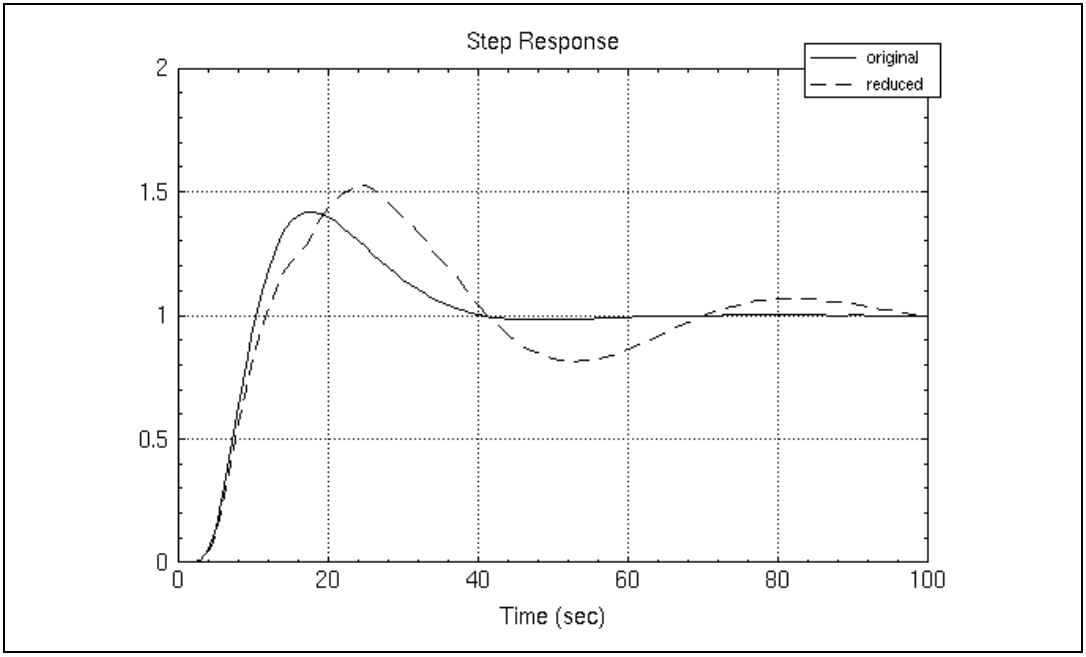

Figure 6-15. Step Response of wtbalance with "input spec"

fracred

`fracred`, the next command examined, has four options—`"right stab"`, `"left stab"`, `"right perf"`, and `"left perf"`.

The options `"left stab"`, `"right perf"`, and `"left perf"` all produce instability. Given the relative magnitudes of the Hankel singular values, this is perhaps not surprising. Figures 6-16, 6-17, and 6-18 illustrate the results using `"right stab"`.

Generate Figure 6-16:

```
svalsrol = svplot(sys*syscr,w,{radians})
plot(svalsol, {keep})
f16=plot(wc,constr,{keep,!grid,
legend=["reduced","original","constrained"],
title="Open-Loop Gain Using fracred()"})
```

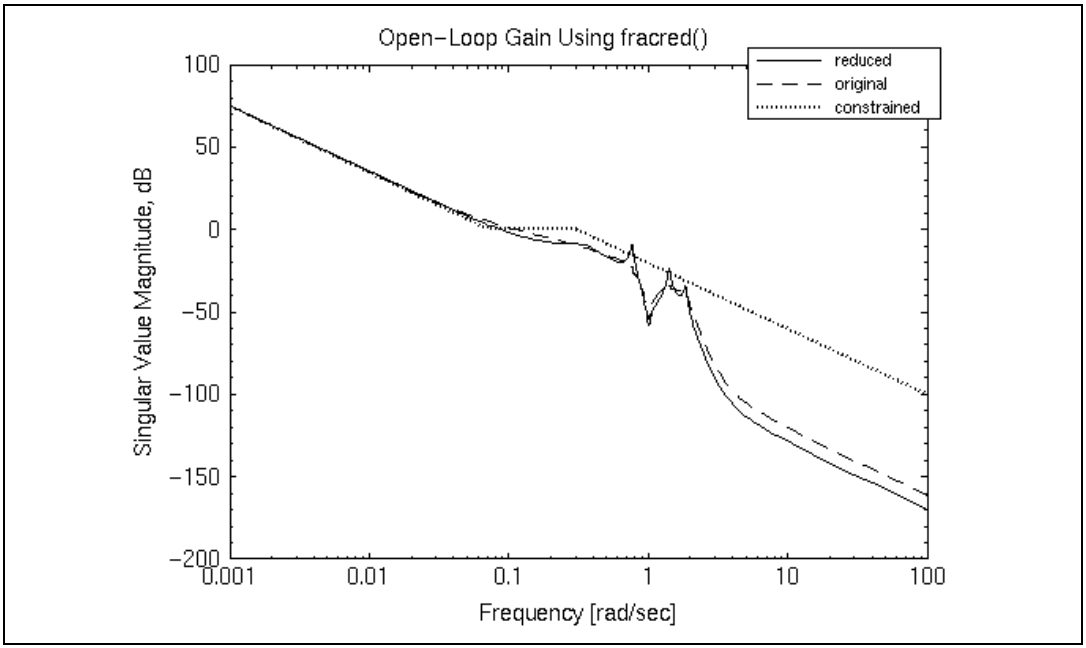

Figure 6-16. Open-Loop Gain Using fracred

Generate Figure 6-17:

```
syscl = feedback(sysol);
sysolr=sys*syscr;
sysclr=feedback(sysolr);
compare(syscl,sysclr,w,{radians,type=5})
f17=plot({keep,legend=["original","reduced"]})
```

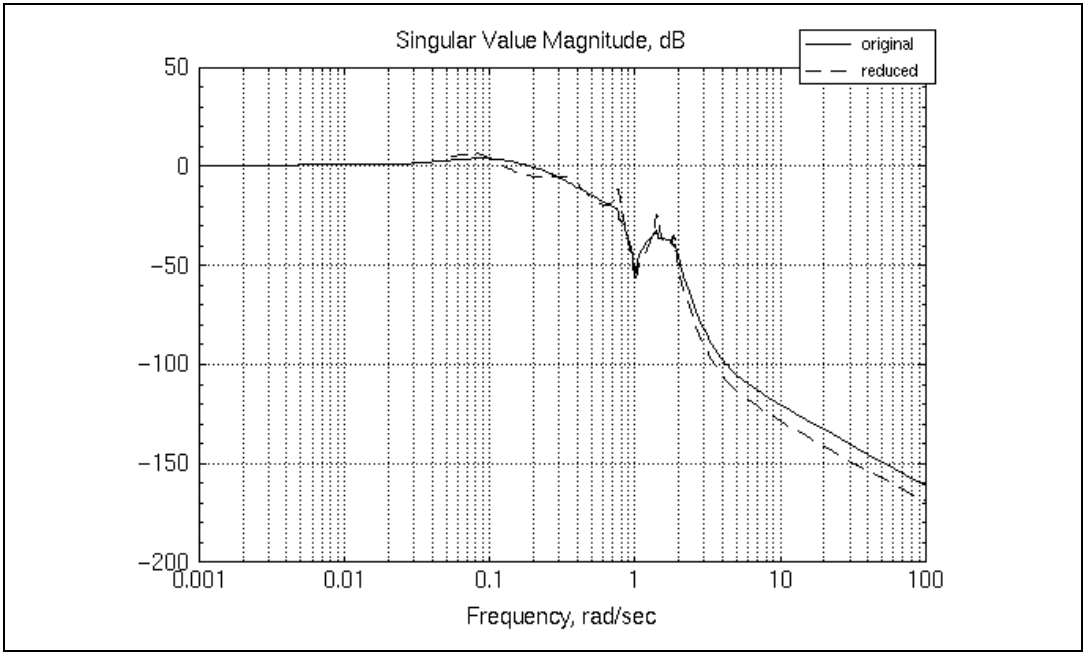

Figure 6-17. Closed-Loop Response with fracred

Generate Figure 6-18:

```
tvec=0:(140/99):140;
compare(syscl,sysclr,tvec,{type=7})
f18=plot({keep,legend=["original","reduced"]})
```

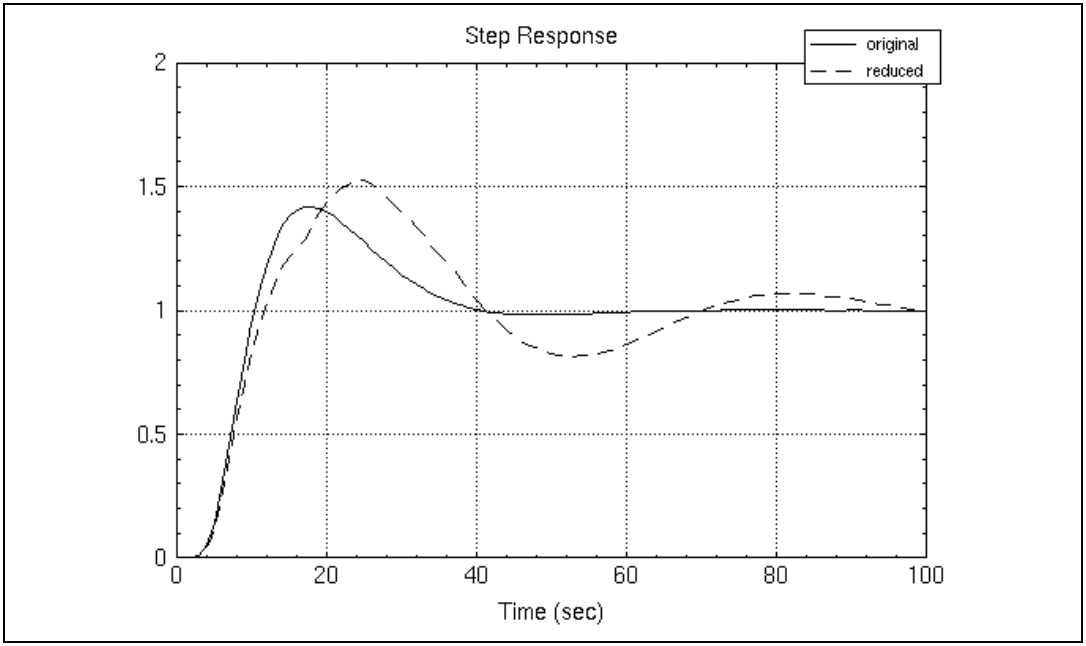

Figure 6-18. Step Response with fracred

The end result is comparable to that from `wtbalance( )` with option `"match"`.

We can create a table to examine the values of the Hankel singular values based on different decompositions approaches.

```
set precision 3  # Optional:
set format fixed # we set a smaller precision here so we
could fit
                 # the table in the manual.
[syscr, hsvrs] = fracred(sys, Kr, Ke, "right stab",2);
[syscr, hsvls] = fracred(sys, Kr, Ke, "left stab",2);
[syscr, hsvrp] = fracred(sys, Kr, Ke, "right perf",2);
[syscr, hsvlp] = fracred(sys, Kr, Ke, "left perf",2);
```

```
hsvtable = [...
"right stab:", string(hsvrs');
"left stab:",  string(hsvls');
"right perf:", string(hsvrp');
"left perf:",  string(hsvlp')]?

hsvtable (a rectangular matrix of strings) =

right stab:3.308  0.728  0.112  0.078  0.024  0.018  0.011  0.010
left stab:1.403  1.331  1.133  1.092  0.965  0.549  0.526  0.313
right perf:0.034  0.016  0.013  0.010  0.004  0.004  0.000  0.000
left perf:4.907  4.874  3.846  3.781  1.225  1.175  0.505  0.041
```

A

Bibliography

[AnJ] BDO Anderson and B. James, "Algorithm for multiplicative approximation of a stable linear system," in preparation.

[AnL89] BDO Anderson and Y. Liu, "Controller reduction: Concepts and approaches," *IEEE Transactions on Automatic Control*, Vol. 34, 1989, pp. 802–812.

[AnM89] BDO Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1989.

[BoD87] S. P. Boyd and J. Doyle, "Comparison of peak and RMS gains for discrete-time systems," *System Control Letters*, Vol. 9, No. 1, pp. 1–6, 1987.

[Enn84] D. F. Enns, "Model reduction with balanced realizations: an error bound and a frequency weighted generalization," *Proceedings for the 23rd IEEE Conference on Decision and Control*, Las Vegas, November 1984, pp. 127–132.

[Enn84a] D.F. Enns, "Model reduction for control systems design," *PhD Thesis*, Dept of Aeronautics and Astronautics, Stanford University, CA, USA, 1984.

[GCP88] K. Glover, R. F. Curtain, and J. R. Partington, "Realisation and approximation of linear infinite-dimensional systems with error bounds," *SIAM J Controls and Optimization*, Vol. 26, 1988, pp. 863–898.

[Glo84] K. Glover, "All optimal Hankel norm approximations of linear multivariable systems and their $L\infty$ error bounds," *Int J Controls*, Vol. 39, 1984, pp. 1115–1193.

[Glo86] K. Glover, "Multiplicative approximation of linear multivariable systems with $L\infty$ error bounds," *Proceedings for American Controls Conference*, Seattle, 1986, pp. 1705–1709.

[GrA86] M. Green and BDO Anderson, "The approximation of power spectra by phase matching," *Proceedings for 25th CDC*, 1986, pp. 1085–1090.

[GrA89] M. Green and BDO Anderson, "Model reduction by phase matching," *Mathematics of Control, Signals, and Systems*, Vol. 2, 1989, pp. 221–263.

[GrA90] M. Green and BDO Anderson, “Generalized balanced stochastic truncation,” *Proceedings for 29th CDC*, 1990.

[Gre88] M. Green, “Balanced stochastic realization,” *Linear Algebra and Applications*, Vol. 98, 1988, pp. 211–247.

[Gre88a] M. Green, “A relative error bound for balanced stochastic truncation,” *IEEE Transactions on Automatic Control*, Vol. 33, 1988, pp. 961–965.

[HiP90] D. Hinrichsen and A J Pritchard, “An improved error estimate for reduced-order models of discrete-time systems,” *IEEE Transactions on Automatic Control*, Vol. 35, 1990, pp. 317–320.

[LAL90] Y. Liu, BDO Anderson, and U-L Ly, “Coprime factorization controller reduction with Bezout identity induced frequency weighting,” *Automatica*, Vol. 26, No. 2, 1990, pp. 233–249.

[Lau80] A. J. Laub, “On computing ‘balancing’ transformations,” *Proceedings on Joint American Controls Conference*, San Francisco, CA, 1980, Section FA8-E.

[LHPW87] A. J. Laub, M. T. Heath, C. C. Paige, and R. C. Ward, “Computation of system balancing transformations and other applications of simultaneous diagonalizing algorithms,” *IEEE Transactions on Automatic Control*, Vol. AC-32, 1987, pp. 115–122.

[LiA86] Y. Liu and BDO Anderson, “Controller reduction via stable factorization and balancing,” *Int. J. Control*, Vol. 44, 1986, pp. 507–531.

[LiA89] Y. Liu and BDO Anderson, “Singular perturbation approximation of balanced systems,” *International Journal of Control*, Vol. 50, 1989, pp. 1379–1405.

[LiA90] Y. Liu and BDO Anderson, “Frequency weighted controller reduction methods and loop transfer recovery,” *Automatica*, Vol. 26, No. 3, pp. 487–489.

[Moo81] B.C. Moore, “Principal component analysis in linear systems: Controllability, observability and model reduction,” *IEEE Transactions on Automatic Control*, Vol. AC-26, No. 1, 1981, pp. 17–32.

[NJB84] C. N. Nett, C. A. Jacobson, and M. J. Balas, “A connection between state-space and doubly coprime fractional representations,” *IEEE Transactions on Automatic Control*, Vol. AC-29, 1984, pp. 831–832.

[PeS82] L. Pernebo, and L. M. Silverman, “Model reduction via balanced state space representations,” *IEEE Transactions on Automatic Control*, Vol. AC-27, No. 2, 1982, pp. 382–387.

[SaC88] M. G. Safonov and R. Y. Chiang, "Model reduction for robust control: a Schur relative-error method," *Proceedings for the American Controls Conference*, 1988, pp. 1685–1690.

[Saf87] M. G. Safonov, "Imaginary-axis zeros in multivariable *H*• optimal control," *Modeling, Robustness, and Sensitivity Reduction in Control*, (Ed. R. F. Curtain), Springer Verlag, Berlin, 1987.

[SCL90] M. G. Safonov, R. Y. Chiang, and DJN Limebeer, "Optimal Hankel model reduction for nonminimal systems," *IEEE Transactions on Automatic Control*, Vol. 35 No. 4, pp. 496–502, 1990.

[Vid85] M. Vidyasagar, *Control Systems Synthesis: A Factorization Approach*, MIT Press, Cambridge, MA, 1985.

[WaS90] W. Wang and M. G. Safonov, "A tighter relative error bound for balanced stochastic truncation," *Systems and Control Letters*, Vol. 14, 1990, pp. 307–317.

[WaS90a] W. Wang and M. G. Safonov, "Comparison between continuous and discrete-time model truncation," *Proceedings for the 29th CDC*, 1990.

[BBK88] S. Boyd, V. Balakrishnan, and P. Kabamba, "A bisection method for computing the L∞ norm of a transfer matrix and related problems," *Mathematical Controls, Signals, and Systems*, Vol. 2, No. 3, pp. 207–219, 1989.

[BeP79] A. Berman and R. J. Plemmons, Nonnegative Matrices in the Mathematical Sciences. *Computer Science and Applied Mathematics series*, Academic Press, 1979.

[BoB90] S. Boyd and V. Balakrishnan. "A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its L∞ norm." *Systems Control Letters* Vol. 15, pp. 1–7, 1990.

[BoB91] S. Boyd and C. Barratt, *Linear Controller Design: Limits of Performance*, Prentice-Hall, 1991.

[BH69] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*, p. 149, Blaisdell Publishing Co., 1969.

[DoS79] J. C. Doyle and G. Stein. "Robustness with Observers," *IEEE Transactions on Automatic Control*, August 1979.

[DoS81] J. C. Doyle and G. Stein. "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis." *IEEE Transactions on Automatic Control*, Vol. AC-26, February 1981, pp 4–16.

[Doy82] J. C. Doyle. “Analysis of Feedback Systems with Structured Uncertainties.” *IEEE Proceedings*, November 1982.

[DWS82] J. C. Doyle, J. E. Wall, and G. Stein. “Performance and Robustness Analysis for Structure Uncertainties,” Proceedings *IEEE Conference on Decision and Control*, pp. 629–636, 1982.

[FaT88] M. K. Fan and A. L. Tits, “m-form Numerical Range and the Computation of the Structured Singular Value.” *IEEE Transactions on Automatic Control*, Vol. 33, pp. 284–289, March 1988.

[FaT86] M. K. Fan and A. L. Tits. “Characterization and Efficient Computation of the Structured Singular Value,” *IEEE Transactions on Automatic Control*, Vol. AC-31, pp. 734–743, August 1986.

[Fr87] B. Francis, *A Course in L. Control Theory*, Springer-Verlag, Berlin-New York, 1987.

[FPGM87] D. S. Flamm, S. Boyd, G. Stein, and S. K. Mitter, “Tutorial Workshop on L∞ Control Theory,” pre-conference workshop, *Proceedings 26th IEEE Conference on Decision and Control*, December 1988.

[GD88] K. Glover and J. C. Doyle, “State-space formulae for all stabilizing controllers that satisfy an L∞ norm bound and relations to risk sensitivity,” *Systems and Control Letters*, Vol. 11, pp. 167–172, 1988.

[DGKF89] J. C. Doyle, K. Glover, P. K. Khargonekar, and B. Francis, “State-space solutions to standard H_2 and L∞ control problems,” *IEEE Transactions on Automatic Control*, Vol. AC-34, No. 8, pp. 831–847, August 1989.

[Gu80] N. K. Gupta, “Frequency Shaping of Cost Functionals: An extension of LQG Design Methods,” *AIAA Journal of Guidance and Control,* Vol. 3, No. 6, December 1980.

[ONR84] *ONR/Honeywell Workshop on Advances in Multivariable Control*, Lecture Notes, Minneapolis, MN, 1984.

[Osb60] E. E. Osborne, “On Preconditioning of Matrices,” *JACM*, 7:338–345, 1960.

[Saf82] M.G. Safonov, “Stability Margins of Diagonally Perturbed Multivariable Feedback Systems,” *IEEE Proceedings*, 129-D:251–256, November 1982.

[SD83] M. G. Safonov and J. C. Doyle. “Optimal Scaling for Multivariable Stability Margin Singular Value Computation,” *Proceedings of MECO/EES 1983, Symposium*, 1983.

[SD84] M. G. Safonov and J. C. Doyle, “Minimizing Conservativeness of Robust Singular Values,” *Multivariable Control*, pp. 197–207, S. G. Tzafestas, ed. D. Reidel Publishing Company, 1984.

[SLH81] M. G. Safonov, A. J. Laub, and G. L. Hartmann, “Feedback Properties of Multivariable Systems: The Role and Use of the Return Difference Matrix,” *IEEE Transactions on Automatic Control*, Vol. AC-26, February 1981.

[SA88] G. Stein and M. Athans. “The LQG/LTR Procedure for Multivariable Control Design,” *IEEE Transactions on Automatic Control*, Vol. AC-32, No. 2, February 1987, pp. 105–114.

[Za81] G. Zames, “Feedback and optimal sensitivity: model reference transformations, multiplicative semi-norms, and approximate inverses,” *IEEE Transactions on Automatic Control*, Vol. AC-26, pp. 301–320, 1981.

[KS72] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*, Wiley, 1972.

B

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at ni.com for technical support and professional services:

- **Support**—Online technical support resources at ni.com/support include the following:
  - **Self-Help Resources**—For immediate answers and solutions, visit the award-winning National Instruments Web site for software drivers and updates, a searchable KnowledgeBase, product manuals, step-by-step troubleshooting wizards, thousands of example programs, tutorials, application notes, instrument drivers, and so on.
  - **Free Technical Support**—All registered users receive free Basic Service, which includes access to hundreds of Application Engineers worldwide in the NI Developer Exchange at ni.com/exchange. National Instruments Application Engineers make sure every question receives an answer.
- **Training and Certification**—Visit ni.com/training for self-paced training, eLearning virtual classrooms, interactive CDs, and Certification program information. You also can register for instructor-led, hands-on courses at locations around the world.
- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program members can help. To learn more, call your local NI office or visit ni.com/alliance.

If you searched ni.com and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

Index

Symbols

*, 1-6
´, 1-6

A

additive error, reduction, 2-1
algorithm
    balanced stochastic truncation (bst), 3-4
    fractional representation reduction, 4-18
    Hankel multi-pass, 2-20
    optimal Hankel norm reduction, 2-15
    stable, 5-2
    weighted balance, 4-12
all-pass transfer function, 1-6, 2-4

B

balance, 1-5, 2-4
    algorithm, 1-11
balanced realization
    definition, 1-10
    internally balanced, 3-9
    singular perturbation, 2-5
    truncation, 2-2, 2-4
balanced stochastic truncation, 3-3
    See also bst
balmoore, 1-5, 2-4
    algorithm, 1-10
bst, 1-5, 1-14, 3-3

C

compare, 1-5, 5-4
controller reduction, 4-2
    with fractional representations, 4-5
controller robustness, 4-2
conventions used in the manual, *iv*

D

diagnostic tools (NI resources), B-1
documentation
    conventions used in the manual, *iv*
    NI resources, B-1
drivers (NI resources), B-1

E

equality bounds, tight, 1-7
error bound, 2-7
    for balanced stochastic truncation, 3-8
    for balanced truncation, 2-2
    for impulse responses, 2-3
    for multiplicative Hankel reduction, 3-16
    for stochastic truncation, 3-9
error formulas, ophank, 2-22
error reduction
    additive, 2-1
    frequency weighted, 1-1, 4-1
    multiplicative, 1-1, 3-1
examples (NI resources), B-1

F

for Hankel norm approximation, 2-7
fracred, 1-5, 4-15
    reduction, 4-18
frequency weighted error reduction, 1-1, 4-1
    controller reduction, 4-2

G

grammians
    controllability, 1-7
    description of, 1-7
    observability, 1-7

H

Hankel matrix, 1-9
Hankel norm approximation, 2-6
Hankel singular values, 1-8, 3-9, 5-1
hankelsv, 1-5, 5-1
    algorithm, multipass, 2-20
help, technical support, B-1

I

instrument drivers (NI resources), B-1
internal balancing, 1-10

K

KnowledgeBase, B-1

L

lmax(A), 1-6
lyapunov, 1-8

M

MATRIXx Help, 1-3
minimality requirements, 1-5
model reduction, schur, 2-5
Moore-Penrose pseudo-inverse, 1-6
mreduce, 1-5, 1-13, 2-10
mulhank, 1-5, 1-14, 3-14
multiplicative error, 1-1, 3-1

N

National Instruments support and
  services, B-1
nomenclature, 1-2
nomenclature for MRM, 1-6
numerical conditioning, 2-8

O

ophank, 1-5, 2-14
    discrete-time systems, 2-21
    error formulas, 2-22
    impulse response error, 2-22
    multipass, 2-20
    onepass, 2-18
    unstable system approximation, 2-23

P

Padé approximation, 1-5
perturbation
    of balanced realization, 2-5
    singular, 1-11, 2-6
phase function, 3-5, 3-15
phase matrix, 3-6, 3-16
pole zero pairs in reduction, 2-4
programming examples (NI resources), B-1

R

redschur, 1-5, 2-4, 2-12
reduced order system, 2-3
Reli(A), 1-6

S

singular perturbation, 1-11
skipChks, 6-3
software (NI resources), B-1
spectral factorization, 1-13
stability requirements, 1-5

stable, 1-5, 5-2
sup, 1-6
support, technical, B-1

T

technical support, B-1
tight equality bounds, 1-7
training and certification (NI resources), B-1
transfer function, allpass, 1-6
troubleshooting (NI resources), B-1
truncate, 1-5, 2-4, 2-11

U

unstable zeros, 2-3

W

Web resources, B-1
wtbalance, 1-5, 4-10